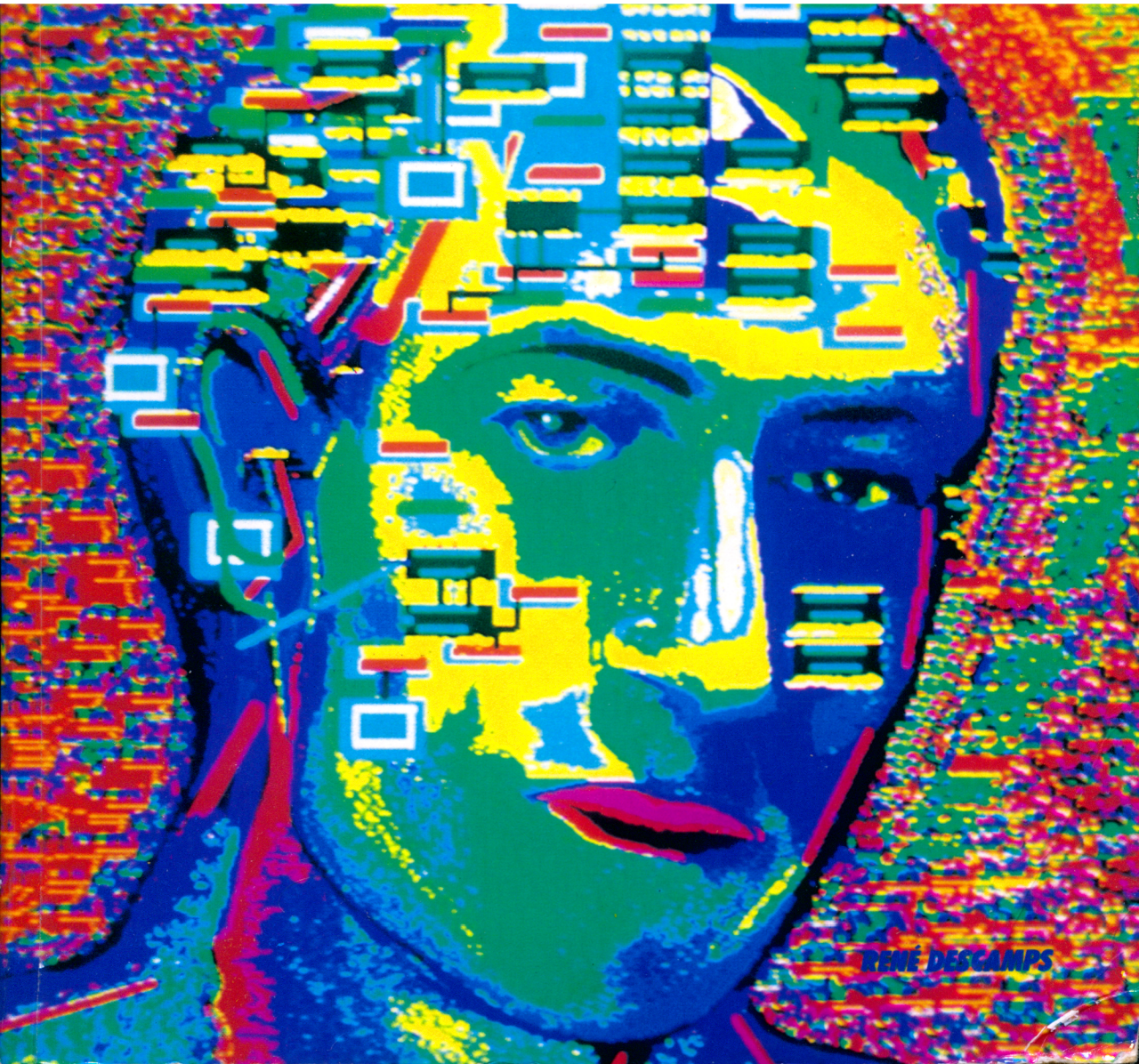
P.S.I.

MICRO ET TECHNIQUE

TROIS ÉTAPES VERS L'INTELLIGENCE ARTIFICIELLE

POUR AMSTRAD CPC



RENÉ DESCAMPS

MICRO ET TECHNIQUE

TROIS ÉTAPES VERS L'INTELLIGENCE ARTIFICIELLE

POUR AMSTRAD CPC

CONNAISSEZ-VOUS TOUTE LA COLLECTION AMSTRAD CHEZ P.S.I. ?

Pour les Amstrad CPC 464, 664 et 6128 :

Initiation :

— La découverte de l'Amstrad - Daniel-Jean David
— Exercices en Basic pour Amstrad - Maurice Charbit

Programmation BASIC :

— 102 programmes pour Amstrad - Jacques Deconchat
— Super jeux Amstrad - Jean-François Sehan
— Amstrad en famille - Jean-François Sehan
— Super générateur de caractères sur Amstrad - Jean-François Sehan
— Photographie sur Amstrad et Apple II - Pierrick Moigneau et Xavier de la Tullaye
— Amstrad en musique - Daniel Lemahieu

Maîtrise du BASIC :

— BASIC Amstrad - 1. Méthodes pratiques - Jacques Boisgontier et Bruno Césard
— BASIC Amstrad - 2. Programmes et fichiers - Jacques Boisgontier
— BASIC Plus, 80 routines sur Amstrad - Michel Martin
— Périphériques et fichiers sur Amstrad - Daniel-Jean David

Assembleur et Pascal :

— Assembleur de l'Amstrad - Marcel Henrot
— Turbo Pascal sur Amstrad - Pierre Brandeis et Frédéric Blanc

Système :

— Clefs pour Amstrad - 1. Système de base - Daniel Martin
— CP/M Plus sur Amstrad 6128 et 8256 - Yvon Dargery
— Clefs pour Amstrad - 2. Système disque - Daniel Martin et Philippe Jadoul

A paraître :

— Graphisme et assembleur sur Amstrad CPC - Francis Piérot
— Intelligence artificielle : langage et formes sur Amstrad - Thierry Lévy-Abégnolli et Olivier Magnan
— Animation graphique sur Amstrad - Gilles Fouchard et Jean-Yves Corre
— Clefs pour Amstrad 8256 - Eric Baumarti

Pour tout problème rencontré dans les ouvrages P.S.I.
vous pouvez nous contacter au numéro ci-dessous :

Numéro Vert/Appel Gratuit en France

05 21 22 01

(Composer tous les chiffres, même en région parisienne)

La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective », et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite » (alinéa 1er de l'article 40).

Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles 425 et suivants du Code Pénal.

© Éditions du P.S.I. - B.P. 86 - 77402 Lagny/Marne cedex
1986

ISBN 2-86595-278-9

MICRO ET TECHNIQUE

TROIS ÉTAPES VERS L'INTELLIGENCE ARTIFICIELLE

POUR AMSTRAD CPC

RENÉ DESCAMPS

ÉDITIONS DU P.S.I.
1986

René Descamps est ancien élève de l'Ecole Polytechnique et Ingénieur en Chef de l'Air (C.R.).

Après SUPAERO et SUPELEC (section radio), ses activités initiales se partagent entre radar et contre-mesures, engins, théorie et applications de la fiabilité des équipements.

Directeur à la SEMA, il mène en parallèle études et applications dans les domaines de la recherche opérationnelle et de la cybernétique. Il participe, à ce titre, à de nombreux congrès internationaux. De cette époque datent ses premiers contacts, toujours maintenus depuis, avec l'Intelligence artificielle.

Il a, ensuite, assuré la Direction générale d'une importante société.

Sommaire

Avant-propos 7

Première partie. – Recherche, adaptation, apprentissage 13

Chapitre 1. – Des programmes qui cherchent et apprennent 15
- Problèmes et solutions 15
- L'Amstrad joue au Taquin 17
- L'Amstrad joue mieux 29
- L'Amstrad joue mieux encore 35
- Reconnaître un chiffre 40

Chapitre 2. – Une idée puissante : la rétroaction 49
- Systèmes autocontrôlés 49
- Stabilité et précision 60
- A la recherche du signe d'une liaison 68
- Universalité des rétroactions 71

Chapitre 3. – L'étrange machine d'Ashby et les autres 73
- L'adaptation des organismes 73
- L'Homéostat d'Ashby et son équivalent sur Amstrad 76
- Les systèmes complexes 82

Chapitre 4. – Structures, mémoire, information 89
- Du bon usage des structures 90
- Mémoire et apprentissage 101
- Recherche et information 111

Deuxième partie. – Jeux de réflexion 119

Chapitre 5. – Jeux dans les arbres 121
- Jeux et Intelligence artificielle 121
- Jeux de réflexion à une personne 123
- Jeux de réflexion à deux personnes 132
- Le MINIMAX 135

Chapitre 6. – Des cailloux et des hommes 145
Wari et solo 145
Création d'un programme de jeu 147
Pour vos débuts, voici AWELE1 149
L'Amstrad s'envole avec AWELE2 162

Chapitre 7. – L'école des dames... et de l'awele 169
Le programme de Samuel 170
Autoformation de l'Amstrad 175
Les frères ennemis 187

Troisième partie. – Systèmes experts 195

Chapitre 8. – Créez vos systèmes experts personnels 197
Préliminaires 197
Votre premier système expert 201
Formez directement vos bases de connaissances avec EXPERS2 215
Un véritable outil d'intelligence artificielle : EXPERS3 222
Pour aller plus loin 233

Chapitre 9. – Des systèmes experts d'un autre type 235
Incertitude et probabilité 235
Un générateur de systèmes experts probabilistes 241
Classification et apprentissage 259

Annexe. – Disquette d'accompagnement 273

Avant-propos

Ne serait-il pas excitant de tenter l'éveil d'une machine aux mouvements de l'esprit ? Ne serait-il pas fascinant de lui poser des questions, de suivre ses démarches dans les labyrinthes tissés par le hasard et l'intelligence, de l'associer à nos jeux de réflexion, de lui demander assistance et de l'obtenir ?

Ce projet des pionniers de l'Intelligence artificielle, rêvé à la mi-temps de notre siècle, n'est certes pas une réalité quotidienne, mais il a déjà pris corps dans plusieurs domaines spécifiques.

Une machine pourrait-elle donc penser ? Question floue, comme le sens des mots, "machine" et "penser", qu'elle assemble. Vous y répondrez vous-même après lecture de ce livre.

En 1950, à l'aube de ce qui n'était pas encore l'informatique, le mathématicien anglais Turing imagina, pour remplacer cette question, un test plus objectif, sous forme d'un jeu d'imitation. Une personne humaine et un ordinateur, également cachés à la vue d'un observateur humain, sont interrogés par celui-ci, à travers écran et clavier, sur tous les sujets qu'il choisira. L'observateur sera-t-il capable de reconnaître qui est l'homme et qui est la machine ?

Si le champ des questions est indéfini, la réponse, à ce jour, est parfaitement positive mais il en irait autrement pour un test de ce type, limité à un jeu de stratégie (échecs, dames, etc.) ou au diagnostic médical, par un système expert, tel MYCIN, dans un domaine restreint. Ce constat est, en soi, assez fantastique.

Comment cela est-il possible ? Ce livre vous invite à le découvrir. Non pas sous forme de théorie ou de récit historique, mais à travers des programmes, une trentaine de programmes, disponibles, en langage BASIC, sur votre micro-ordinateur et dûment commentés.

Chacun d'eux vous propose une expérience singulière. L'un résout des puzzles au Taquin, d'autres simulent des pilotes automatiques pour atterrissage en vol vertical ou conduite de navires. Une étrange machine réagit aux perturbations en réorganisant d'elle-même sa structure interne. Vous bâtirez de toutes pièces un programme jouant à l'awele (jeu africain). Il pourrait se révéler un partenaire difficile,

d'autant qu'il est capable d'améliorer encore son jeu à votre contact. Si vous le négligez, il se dédoublera et jouera contre lui-même.

Une batterie de systèmes experts vous attend pour créer et utiliser les "bases de connaissance" de votre choix. Certaines sont déjà construites et prêtes à l'emploi. Vous pourrez ainsi élaborer un programme de vacances nautiques, déterminer votre caractère personnel et ceux de vos amis, répondre aux questions choisies d'un programme de diagnostic médical qui listera les affections dont, à son avis, vous souffrez et leurs probabilités. Vous pourrez même tenter de lier les qualités et défauts de vos amis à leur signe zodiacal.

Le champ de l'Intelligence artificielle est devenu si vaste qu'il serait malaisément couvert en un seul ouvrage. Aussi avons-nous retenu un thème central développé en trois parties.

Thème central : la résolution de problèmes définis par le but visé, la finalité.

Toute discipline ou technique traite un certain type de problèmes et tente de les résoudre. Mais, dans les situations classiques, ces disciplines privilégient la création préalable, par l'intelligence humaine, d'outils de solution. Leur idéal est l'élaboration, puis l'usage, de méthodes prédéfinies, systématiques et sûres, en un mot, d'algorithmes.

Différente est l'approche de l'Intelligence artificielle. Simplifions pour mieux nous faire comprendre : seuls sont donnés une situation et le résultat désiré. La méthode de solution fait partie des inconnues. Cela implique la recherche dans un espace de configurations, l'exploration de cet espace, l'adaptation aux diverses situations rencontrées, un éventuel apprentissage qui tire profit de l'expérience acquise.

La recherche, certes, ne peut être laissée au hasard. Elle est guidée par des stratégies générales : méthodes pour découvrir des méthodes, stratégies qui s'inspirent des procédés de l'intelligence humaine qu'elles simulent alors, d'où dérivent de nouveaux concepts tenant compte de la structure spécifique des ordinateurs.

Leur trait commun : tirer le meilleur parti des seules données connues :

– le but à atteindre ;
– la structure de l'espace à explorer.

Les notions de "feedback" ou rétroaction (recherche guidée par l'écart entre la situation actuelle et le but) et la mise en œuvre d'objectifs intermédiaires, dérivés du but final, vont donc jouer, ici, un rôle essentiel.

La première partie de ce livre (chapitres 1 à 4) a pour titre "Recherche, adaptation, apprentissage". Elle vise à illustrer et développer de façon concrète les notions précédentes.

Le chapitre 1 introduit le sujet par deux exemples, complètement traités sous forme de quatre programmes. Premier thème : l'ordinateur joue au Taquin et doit résoudre les puzzles qui lui sont présentés, sans perdre ses octets dans le magma des 362 880 configurations possibles. Trois stratégies, de complexité et d'efficacité croissantes, s'incarnent dans trois programmes successivement construits. Un autre problème ouvre la voie à une technique simple d'apprentissage : l'Amstrad doit reconnaître les symboles graphiques des chiffres arabes et les associer aux nombres correspondants.

La rétroaction ou feedback est l'idée centrale du chapitre 2. Celui-ci débute par deux courts programmes de simulation, relatifs, le premier à un pilote automatique d'engin en atterrissage vertical, le second à un "ogre" à tête chercheuse. Le programme CYBER, simulation d'un pilote automatique agissant sur la dérive d'un navire, illustre les notions de stabilité, précision, prévision et démontre, par le tracé des courbes de réponse sous différents réglages, l'importance cruciale de ceux-ci. Des situations plus complexes exigent une analyse fine des réactions entre commandes et résultats et donc une recherche exploratoire. Deux programmes simples, baptisés ZÉRO et OPTIMUM, mettent en œuvre ces méthodes.

La vedette du chapitre 3 est l'HOMEOSTAT du cybernéticien anglais Ross Ashby. Cette machine bizarre se plaît dans une certaine configuration d'équilibre. Ecartée par force de celle-ci, elle est capable de modifier ses liaisons internes jusqu'à l'équilibre retrouvé, mais dans une structure différente. Quel en est l'intérêt ? Simuler certains modèles d'adaptation des êtres vivants, dans des situations qui dépassent les possibilités des mécanismes de feedback. Le programme ASHBY fournit une version électronique de cette machine et permet d'en observer les réactions. La suite du chapitre prouve, par l'exemple des automates cellulaires, combien il est facile de créer des systèmes très complexes. Ils peuvent simuler n'importe quel ordinateur et leur comportement n'est pas calculable. Certaines applications sont indiquées. De plus, grâce au programme CELLULES, votre écran pourra se couvrir de beaux graphismes.

Au chapitre 4, nous tentons un essai sur un sujet difficile, au centre de cette discipline récente appelée cybernétique. A vous de juger si la transformation est, ou non, réussie. Premier thème : l'influence de la "structure" d'un système sur la difficulté, plus ou moins grande, de solution des problèmes qui s'y rapportent. Ainsi se distinguent des systèmes séparables, ordonnés, hiérarchisés... Le programme LOTERIE, montre, sur un exemple très simple, les avantages des systèmes séparables... ou presque séparables. Mais, au-delà des structures, la mémoire et l'expérience acquise déterminent la puissance et l'efficacité des solutions. Dans ce contexte, le programme MÉMOIRE propose un noyau de système expert, capable d'apprentissage en tous domaines. L'exemple choisi d'application, l'éducation d'un oisillon, glissera, je l'espère, un brin de fantaisie dans les pages de ce chapitre. Autre sujet

important, lié à l'apprentissage : comment poser les questions efficaces et, à cette fin, peut-on mesurer leur efficacité ? La belle théorie de l'information, due à Shannon, apporte des réponses dans certains cas. Nous essayons d'en donner une idée par des exemples simples (deviner un mot ou la combinaison d'un coffre-fort) et un mini-programme : SHANNON.

Deuxième partie : "Jeux de réflexion" ou de stratégie. Comment un ordinateur peut-il résoudre des puzzles (jeux à une personne) ou tenir honorablement sa place dans un jeu de réflexion à deux personnes, échecs, dames, checkers, awele, etc ? Le chapitre 5 répond à ces questions. Un aperçu historique montre la place essentielle tenue par les jeux de réflexion au berceau de l'Intelligence artificielle. Herbert Simon, prix Nobel d'Economie, n'écrivait-il pas, en 1963 (c'était hier !) : "Construire une machine qui joue avec succès aux échecs signifierait, semble-t-il, pénétrer au cœur des activités intellectuelles humaines".

Les méthodes et principes qui ont permis ce succès comprennent les stratégies générales, décrites dans la première partie de cet ouvrage et des techniques précises développées ici sous forme de logiciels travaillant sur des "graphes", représentations symboliques des jeux réels. Les deux programmes JEU1 et JEU2 développent des solutions utilisables pour les puzzles. MINIMAX met en œuvre l'algorithme du même nom pour les jeux à deux personnes. Mais le progrès le plus notable et le plus récent en ce domaine est lié à l'algorithme dit "Alpha-Beta" et à ses diverses variantes. Le programme ALBE et les commentaires qui l'accompagnent vous en diront davantage.

Vous souhaitez sans doute des programmes qui soient de rudes partenaires mais que vous puissiez décortiquer pour en connaître les secrets. Le chapitre 6, intitulé "Des cailloux et des hommes", vous en offre deux : AWELE1 et AWELE2. Nous avons choisi cet africain, représentant d'une famille de jeux passionnants, raffinés et d'une grande profondeur stratégique. Vous pourrez vous initier, si ce n'est déjà fait, avec AWELE1, vous battre à armes égales avec AWELE2 et ensuite, à votre gré, expérimenter, modifier, améliorer.

Ne serait-ce pas mieux encore si votre partenaire de métal et de silicium était capable de se perfectionner, par ses propres moyens, au cours de vos joutes homériques ? Ce n'est pas une mince affaire mais pourquoi ne pas essayer ? Voyons d'abord comment s'y est pris Samuel, un grand ancêtre, avec son programme de checkers (les dames américaines). Ce dernier (le programme) se hissa, à la force du poignet, à un rang très honorable dans la hiérarchie checkérienne et battit, voici trente ans déjà, plusieurs champions. AWELEVOL vous propose, sur d'autres idées, un modeste équivalent. Il est capable, au cours d'une même partie, d'apprendre, à chaque coup joué, quelque chose de nouveau et de modifier sa technique en conséquence.

Nous pouvons, d'une autre façon, juger des résultats obtenus en modifiant les paramètres du programme. Il suffit de le laisser jouer

contre lui-même, chacun des partenaires étant différemment équipé. Lançons le programme NARCISSE et que le meilleur gagne ! Le spectacle en vaut la peine et nous y apprendrons sans doute quelque chose. Si d'ailleurs vous en avez le goût, pourquoi ne pas aller plus loin ? Créez une boucle supplémentaire de rétroaction qui modifiera automatiquement les coefficients des deux partenaires à mesure de leurs succès ou échecs. Partez alors tranquillement en vacances et laissez vos champions s'échiner par Amstrad interposé. Vous ramasserez les marrons au retour.

Les systèmes experts sont à la mode et ce pour de bonnes raisons. Le chapitre 8 vous invite à créer vos systèmes experts personnels. Nous verrons d'abord à quoi servent ces drôles de machines, pourquoi et dans quel contexte elles vinrent au monde, à quoi elles ressemblent et quelles en sont les applications principales.

Vous créerez ensuite votre premier (sans doute ?) programme de ce type : EXPERS1 puis ses variantes plus performantes : EXPERS2 et EXPERS3. Chacun d'eux est capable de lire un nombre illimité de "bases de connaissances", liant des conditions (critères et valeurs) et des réponses. Chaque élément de la base associe plusieurs conditions (liées par la conjonction ET) à une ou plusieurs réponses. Il vous suffit ensuite de décrire une situation particulière et d'exprimer vos souhaits. Le génie d'Aladin vous renseignera. Bien sûr, il convient, au préalable, de construire ces bases de connaissances. C'est facile avec des listes de DATA. Vous en trouverez déjà deux, toutes faites, pour essayer les logiciels. L'une est relative au choix d'un langage de programmation. L'autre pourra vous être réellement utile si vous appréciez les sports nautiques en eau douce (House-boat, voile, planche à voile, kayak, raft). Choisissez un sport, une région de France et exprimez ces choix. Vous obtiendrez en retour, immédiatement, la liste des fleuves, lacs et rivières qui pourraient vous convenir.

La vie est, hélas (ou heureusement), pleine d'incertitudes. Innombrables sont les questions auxquelles une réponse par oui ou par non est tout simplement impossible. Nos systèmes experts ne peuvent ignorer, plus longtemps, ces dures réalités. Allez donc vite au chapitre 9, dernier du nom. EXBAYES vous y attend. Comme son titre l'indique peut-être, c'est un système expert probabiliste, fondé sur le théorème, déjà ancien, du Révérend Bayes. Le programme se repaît, comme ses confrères, de bases de connaissances, mais, pour celles-ci, réponses et conditions sont liées par des probabilités. Pour vous familiariser avec EXBAYES et peut-être vous distraire, nous avons prévu deux bases : CARACT et DOCTEUR.

La première vous livrera le caractère ou tempérament secret de tel(le) ou tel(le) ami(e), préalablement soumis(e) à un petit questionnaire. Vous saurez ainsi s'il (si elle) est apathique, nerveux(se), sentimental(e), passionné(e) etc. Vous pourrez toujours vérifier en effeuillant une marguerite.

La seconde effectue, selon vos réponses à ses intelligentes questions, un diagnostic médical limité (espace oblige) aux voies respiratoires. Pour les autres types d'affections (et même pour les précédentes) veuillez consulter votre médecin quotidien.

Maintenant (c'est devenu un leitmotiv !), pourquoi pas un système expert capable d'apprendre ? Voici, pour vous servir, EXCLASS, extension du programme MÉMOIRE, rencontré au chapitre 4. EXCLASS travaille, sur tout sujet de votre choix, avec deux listes : celle des critères et celle des classes ou catégories. Décidez d'un thème et discutez-en quelques instants avec lui. Proposez-lui des situations et demandez leur classe d'appartenance. Il répondra et commettra, au départ, nombre d'erreurs. Rectifiez-les patiemment et votre obligeance sera récompensée. EXCLASS finira par tout savoir dans le domaine choisi. Pour vous en convaincre, nous avons construit la base CARASTRO. Les critères sont des traits de caractère, défauts ou qualités. Les classes sont les signes du Zodiaque. Enseignez à EXCLASS quelques notions élémentaires d'astrologie (incluses, sans frais, dans le chapitre 9). Décrivez-lui alors la personnalité d'une de vos relations. Il devrait (sans garantie de l'auteur ni de l'éditeur) pouvoir deviner sous quel signe est née cette personne.

Si le domaine des systèmes experts vous intéresse et que vous souhaitiez en savoir davantage, je me permets de vous suggérer la lecture de *Apple, logique et systèmes experts,* du même auteur, publié en 1985 aux Editions du P.S.I.

Un dernier mot : les trois parties de ce livre peuvent être lues dans l'ordre qui vous conviendra mais, à l'intérieur de chaque partie, il est préférable de respecter le rang des chapitres composants.

Il m'est très agréable de clore cet avant-propos par des remerciements à :

Philippe Descamps dont les remarques m'ont aidé à préciser la structure de l'ouvrage ;

Nitsan Seniak et Augustin Garcia qui ont beaucoup contribué à la présentation graphique des programmes de la première partie.

PREMIÈRE PARTIE
RECHERCHE, ADAPTATION, APPRENTISSAGE

Chapitre 1
DES PROGRAMMES QUI CHERCHENT ET APPRENNENT

PROBLÈMES ET SOLUTIONS

Le thème central de ce livre, avons-nous dit en préambule, est lié à la solution de problèmes définis par le but à atteindre. Mais qu'est-ce qu'un problème ? Pouvons-nous préciser les composants d'une telle notion ? Après analyse, nous distinguerons :

- un espace des **configurations** ou encore des **solutions possibles,** muni d'une structure définissant, en particulier, les cheminements autorisés ;
- un **générateur** de telles configurations capable d'explorer cet espace selon des voies à définir ;
- un **vérificateur** assorti des conditions qui permettent de contrôler si un élément choisi de l'espace précédent est, ou non, une **solution réelle** du problème posé ;
- une **stratégie de recherche** ou **heuristique** qui puisse orienter ou réorienter l'exploration de l'espace des configurations selon la "distance" entre objectif visé et configuration en cours d'examen.

Mais une telle nomenclature est bien abstraite. Aussi semble-t-il plus clair et plus efficace d'illustrer ces notions par un exemple complètement traité et, de préférence, sous forme de programmes. Nous avons choisi le jeu de Taquin.

Les méthodes d'apprentissage seront ensuite abordées grâce à un exemple simple : instruire l'Amstrad à reconnaître les formes graphiques représentant les chiffres arabes.

Une version bien connue du Taquin utilise neuf cases, disposées en carré, occupées par des chiffres de 1 à 8 et par une case vide. Grâce à cette dernière, les chiffres qui en sont voisins peuvent être déplacés parallèlement aux bords du carré.

A partir d'une configuration initiale telle que la suivante :

5	3	2
1	7	6
4		8

Fig. 1-1

Il s'agit d'atteindre une position finale plus ordonnée, par exemple :

1	2	3
4		5
6	7	8

Fig. 1-2

L'espace des solutions possibles est formé des diverses configurations. Leur nombre est la factorielle de 9, soit 362880. A partir de l'une d'entre elles, on peut accéder aux positions filles, au nombre de :

- quatre si la case vide est centrale ;
- trois si la case vide occupe un milieu de côté ;
- deux si la case vide occupe un coin.

Construire, de proche en proche, ces solutions possibles est la tâche du générateur. Les positions successives peuvent être figurées (avec beaucoup de temps et de papier), par les sommets d'un graphe. Un sommet est relié aux successeurs (ou positions filles) par des arcs orientés *(voir la figure 1-3 ci-contre).*

Le graphe peut comporter des circuits (retour à une position déjà utilisée) car il est assez facile de tourner en rond. Le vérificateur s'assurera qu'une configuration atteinte est, ou non, identique à la configuration finale souhaitée.

Au cours de ce chapitre seront explorées trois stratégies de recherche. Chacune engendrera un programme spécifique capable, le plus souvent, de trouver la solution d'un puzzle au Taquin.

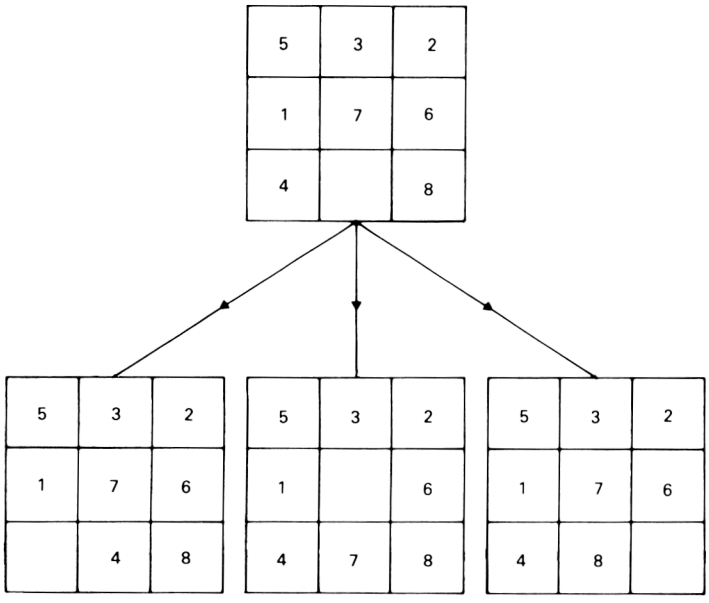
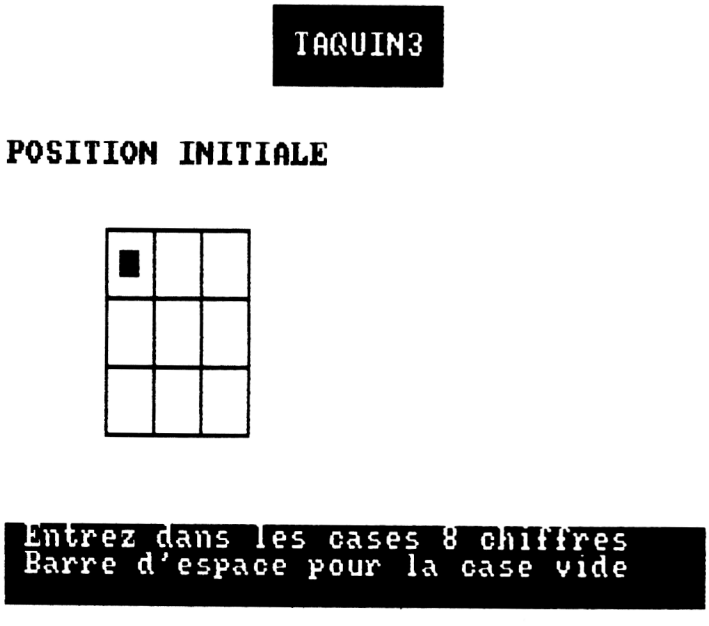


Fig. 1-3

L'AMSTRAD JOUE AU TAQUIN

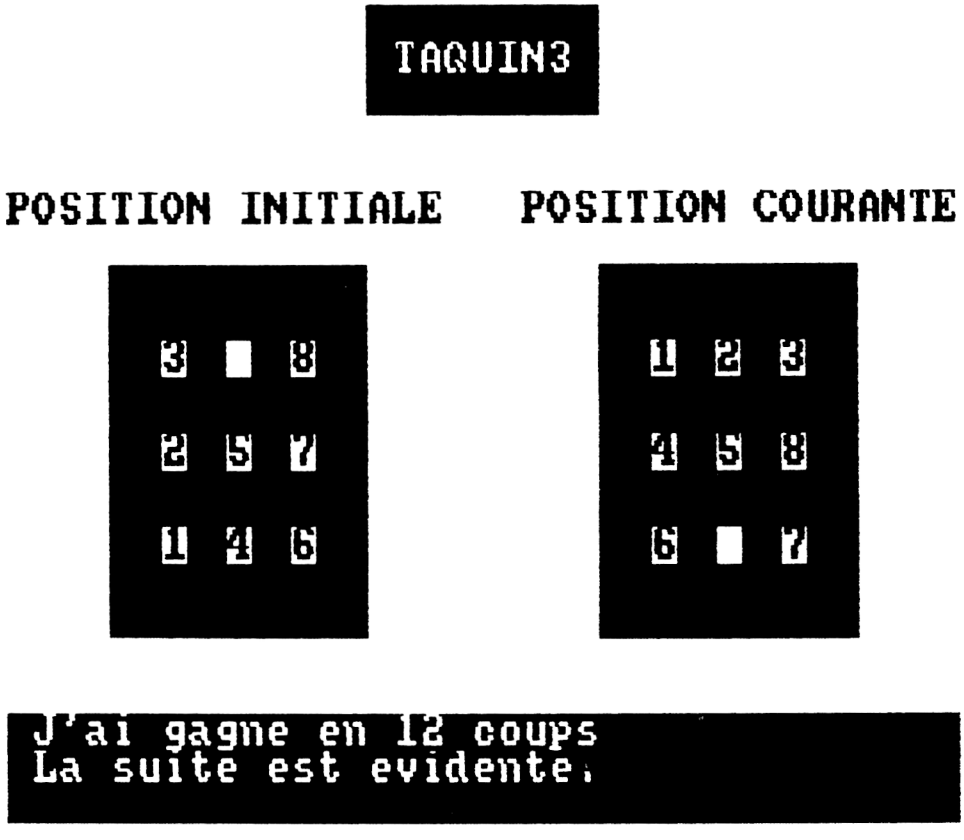
Vous venez de transcrire, sur cassette ou disquette, le programme TAQUIN3. Vous le chargez et le lancez. L'écran affiche la grille ci-dessous.



Entrez, au clavier, une configuration, par exemple, la suivante :

3		8
2	5	7
1	4	6

Les chiffres se déplacent sur la grille de droite, pendant deux minutes environ, jusqu'à l'affichage final, accompagné de deux notes triomphantes.



Vous déplacez mentalement le 8 puis le 5 pour constater que le programme a réussi. Un résultat analogue serait obtenu en 6 coups avec une position initiale telle que :

5		2
1	7	3
4	6	8

Pour être honnête, à partir d'une configuration quelconque, le résultat ne sera pas toujours aussi brillant. Le problème peut n'avoir aucune solution et, s'il en a une, plusieurs dizaines de coups seront généralement nécessaires. Mais ces nombres restent modestes comparés aux 362 880 (soit 9 !) qui constituent le domaine de recherche. Comment parvenir à ce résultat ?

La mise en scène

Il nous faut d'abord représenter toute configuration possible sous une forme compacte (attention aux temps de calcul !). Pour ce faire, après avoir numéroté les 9 cases de la grille dans cet ordre :

1	2	3
4	5	6
7	8	9

nous choisirons comme support une simple chaîne de caractères à 9 positions. Par exemple, la configuration :

3		8
2	5	7
1	4	6

sera mémorisée dans la chaîne :

CF$="398257146"

où le 9 symbolise la case vide. Une table CF$(1), de dimension 4, permettra alors de stocker une position initiale, soit CF$(0), et ses successeurs (4 au maximum).

Nous devons ensuite définir les "configurations filles" pouvant succéder à une configuration donnée, selon la position I de la case vide. Une table de chaînes de caractères SC$(I), de dimension 9, fera l'affaire. Elle contiendra :

- le nombre de successeurs,
- les repères, dans la grille, des cases adjacentes.

Par exemple, la chaîne SC$(5) aura la forme :

SC$(5)="42468"

ce qui veut dire, pour une case vide au centre (repère 5) :
- 4 successeurs,
- cases adjacentes : 2,4,6,8.

Nous aurons également besoin, notamment pour calculer la "distance" entre deux configurations, de transformer le schéma linéaire d'une position en une grille sur laquelle une case de rang I sera définie par ses deux coordonnées dans le système d'axes U,V.

V \ U	1	2	3
1	1	2	3
2	4	5	6
3	7	8	9

D'où deux nouvelles tables U(I) et V(I) de dimensions 9. Par exemple, à la case 4 seront associées les coordonnées U=1,V=2. Les tables SC$(I), U(I), V(I) seront remplies avec des DATA, selon les premières instructions du programme, listées ci-dessous :

```
10 DEFINT a-z
20 MODE 1:PEN 0:PAPER 13:CLS:PEN#3,0:PEN
#4,0
30 WINDOW#3,17,25,3,5:PAPER#3,0:CLS#3
40 WINDOW#3,16,24,2,4:PAPER#3,10:CLS#3
50 LOCATE#3,2,2:PRINT#3,"TAQUIN1"
60 WINDOW#3,7,16,10,19:PAPER#3,0:CLS#3
70 WINDOW#3,6,15,9,18:PAPER#3,10:CLS#3
80 WINDOW#4,3,39,22,24:PAPER#4,0:CLS#4
90 WINDOW#4,2,38,21,23:PAPER#4,10:CLS#4
100 DIM cf$(4),sc$(9),in(4),ch(4),u(9),v
(9)
110 RANDOMIZE TIME
120 '      Positions suivantes
130 DATA 224,3135,226,3157,42468
140 DATA 3359,248,3579,268
150 FOR i=1 TO 9
160    READ sc$(i)
170 NEXT i
180 '      Solution finale
190 FOR i=1 TO 9
200   READ u(i),v(i)
210 NEXT i
220 DATA 1,1,2,1,3,1
230 DATA 1,2,3,2,1,3
240 DATA 2,3,3,3,2,2
```

Les instructions graphiques, fort agréables, du BASIC Amstrad, sont mises en œuvre dans les sous-programmes "GRILLE" et "ENTREE" pour tracer, sur l'écran, une grille relocalisable et construire la configuration initiale.

Un brin de stratégie

Les choses sérieuses commencent. Quelles indications allons-nous donner au programme pour guider sa recherche vers la configuration terminale ?

L'idée la plus simple serait de cheminer au hasard. Pour éviter le risque permanent de tourner en rond, il conviendrait de noter et de garder en mémoire les positions déjà rencontrées et les chemins

parcourus. Si nous retombons dans une telle position, il faudrait en repartir dans une direction différente. Beaucoup de mémoire et de temps, pour quels résultats ? Une parabole nous aidera à mieux faire.

Nous nous situons sur un plateau très bosselé et creusé de multiples voies. Nous devons trouver notre chemin vers la mer invisible au niveau zéro. La seule information disponible est l'altitude de chaque position à la croisée de chemins (2, 3 ou 4 selon le cas) qui mènent à d'autres carrefours. A partir de notre position instantanée, nous pouvons emprunter tous les chemins qui s'offrent, jusqu'à la prochaine croisée, et noter l'altitude de chaque point d'arrivée. Nous choisirons la voie qui mène à l'altitude intermédiaire **minimum.** Elle ne sera pas forcément inférieure à l'altitude de départ si toutes les voies utilisables sont ascendantes. Enfin, pour départager les éventuels *ex-æquo,* nous choisirons au hasard.

A priori, cette méthode offre de bonnes chances d'atteindre notre but. Nous n'éviterons pas, chemin faisant, d'entrer dans des boucles mais les choix, au hasard, des voies d'altitudes égales nous en feront sortir un jour ou l'autre.

Transformer cette idée en programme exige de donner un sens à la notion d'altitude puis de trouver un indicateur calculable et efficace. Ce sera la "distance" entre une configuration quelconque et la configuration terminale. A la réflexion se proposent plusieurs indicateurs de cette sorte. Leur efficacité sera surtout mesurée à l'expérience. Citons :

- le nombre de chiffres mal placés ;
- la somme des distances "MANHATTAN" de tous les chiffres de la configuration étudiée. Pour chaque chiffre, elle sera donnée par le nombre de mouvements nécessaires sur la grille, parallèlement aux axes de celle-ci, pour faire coïncider les positions initiale et finale de ce chiffre (MANHATTAN se réfère évidemment à la structure en lignes parallèles et perpendiculaires des rues de ce quartier de New York) ;
- la somme des carrés de telles distances ;
- la somme des puissances N (pour N pair) ;
- ces mêmes sommes, pondérées selon les chiffres par des coefficients à définir ;
- toute autre fonction positive imaginable des distances individuelles.

Il faut bien se décider. Notre choix, que le lecteur (ou la lectrice) est vigoureusement invité(e) à remettre en cause, est guidé par les considérations suivantes :

- l'indicateur doit être le plus sélectif possible. Cette condition sera d'autant mieux vérifiée, semble-t-il, que le domaine des valeurs prises par la fonction qui le représente sera plus étendu ;
- un poids particulier peut être donné à un chiffre (parmi les 8) afin qu'il atteigne plus vite sa position finale.

Selon ces présomptions, nous avons retenu :

- la somme des puissances quatrièmes des déplacements dX et dY selon les axes de la grille ;
- un facteur 16 pondérant l'indicateur relatif au chiffre 1, contre un coefficient égal à 1 pour les autres chiffres.

Le cœur du programme TAQUIN1

La mise en forme des deux fonctions de base, définition des successeurs et calcul de la "distance" est réalisée par les deux sous-programmes que voici :

```
560 '
570 '       SUCCESSEURS
580 '
590 z$=cf$(0):j0=INSTR(z$,"9"):c$=LEFT$(
sc$(j0),1):ns=VAL(c$)
600 FOR i=1 TO ns
610   z$=cf$(0):c$=MID$(sc$(j0),i+1,1):j
1=VAL(c$):ch$=MID$(z$,j1,1):ch(i)=VAL(ch
$)
620   MID$(z$,j0,1)=ch$:MID$(z$,j1,1)="9
":cf$(i)=z$
630 NEXT
640 RETURN
650 '
660 '       DISTANCE MANHATTAN
670 '
680 dm=0
690 FOR j=1 TO 9
700   c$=MID$(cf$(i),j,1):ch=VAL(c$):x=(
j-1) MOD 3+1:y=(j-1)\3+1
710   dx=u(ch)-x:dy=v(ch)-y
720   IF ch=1 THEN ft=16 ELSE ft=1
730   dm=dm+ft*(dx*dx*dx*dx+dy*dy*dy*dy)
740 NEXT j
750 RETURN
```

Les entrées et les sorties correspondantes peuvent être ainsi listées :

	Entrées	*Sorties*
Successeurs	Position de départ : CF$(0)	Repère de la case vide : J0 Nombre de successeurs : NS Chiffres pouvant être déplacés : CH(I) Configurations filles : CF$(I)
Distance	Configuration fille : CF$(I)	Distance associée : DM

On notera (ligne 700) les formules qui permettent le calcul des coordonnées X, Y d'une case de la grille à partir du rang J de cette case. Elles utilisent les fonctions MOD et \ de la division entière.

La stratégie définie est traduite dans une boucle de recherche (instructions 390-550). Pour chaque configuration fille CF$(I), la distance mesurée entre dans une table d'indicateurs IN(I). Leur minimum MI est calculé ainsi que l'indice I0 de la configuration fille associée à ce minimum. S'il n'apparaît pas de minimum strict, le choix I0, comme déjà indiqué, est tiré au sort entre les valeurs égales. Le chiffre déplacé est mémorisé dans la variable Z0 afin d'éviter les marches arrière.

Enfin le programme affiche le mouvement calculé.

```
390 '
400 '------ Boucle de recherche
410 '
420 ne=ne+1:GOSUB 590
430 FOR i=1 TO ns:GOSUB 680:in(i)=dm:NEX
T
440 mi=1000
450 FOR i=1 TO ns
460    IF ch(i)=z0 THEN 490
470    IF in(i)<mi THEN mi=in(i):i0=i:GOT
O 490
480    IF in(i)=mi THEN IF RND<0.5 THEN i
0=i
490 NEXT
500 z0=ch(i0):GOSUB 1090
510 LOCATE#4,12,2:PRINT#4,STRING$(3," ")
520 LOCATE#4,12,2:PRINT#4,mi:LOCATE#4,35
,2:PRINT#4,z0
530 IF mi=0 THEN CLS#4:LOCATE#4,2,2:PRIN
T#4,"J'ai gagne en";ne;"coups.":SOUND 1,
478,50:SOUND 1,379,50:SOUND 1,319,100:LO
CATE 1,1:END
540 cf$(0)=cf$(i0)
550 GOTO 420
```

Liste des variables

Le programme complet, précédé d'une liste des variables, est proposé ci-après.

Tables :

CH(4) : chiffres pouvant être déplacés.
CF$(4) : représentation d'une configuration et des configurations "filles".

IN(4) : indicateurs attachés aux configurations.
SC$(9) : mouvements autorisés selon la position de la case vide.
U(9),V(9) : tables définissant la solution finale.

Variables simples :

C$: caractère courant.
CH : mémorisation d'un chiffre.
Ch$: variable intermédiaire.
DEL : maille de la grille.
DM : distance Manhattan.
DX,DY : écarts de position.
FT : facteur de pondération.
I,J : indices de boucle.
I0 : indice de la position choisie.
J0,J1 : indices intermédiaires.
MI : minimum des indicateurs.
NE : nombre d'essais.
NS : nombre de successeurs.
R$: entrée d'un chiffre au clavier.
X,Y : coordonnées dans la grille et indices de boucle.
X0,Y0 : origine de l'écran graphique.
Z0 : mémorisation d'un chiffre déplacé au coup précédent.
Z$: variable intermédiaire pour une configuration.

Listing du programme TAQUIN1

```
10 DEFINT a-z
20 MODE 1:PEN 0:PAPER 13:CLS:PEN#3,0:PEN
#4,0
30 WINDOW#3,17,25,3,5:PAPER#3,0:CLS#3
40 WINDOW#3,16,24,2,4:PAPER#3,10:CLS#3
50 LOCATE#3,2,2:PRINT#3,"TAQUIN1"
60 WINDOW#3,7,16,10,19:PAPER#3,0:CLS#3
70 WINDOW#3,6,15,9,18:PAPER#3,10:CLS#3
80 WINDOW#4,3,39,22,24:PAPER#4,0:CLS#4
90 WINDOW#4,2,38,21,23:PAPER#4,10:CLS#4
100 DIM cf$(4),sc$(9),in(4),ch(4),u(9),v
(9)
110 RANDOMIZE TIME
120 '      Positions suivantes
130 DATA 224,3135,226,3157,42468
140 DATA 3359,248,3579,268
150 FOR i=1 TO 9
160    READ sc$(i)
170 NEXT i
180 '      Solution finale
```

```
190 FOR i=1 TO 9
200   READ u(i),v(i)
210 NEXT i
220 DATA 1,1,2,1,3,1
230 DATA 1,2,3,2,1,3
240 DATA 2,3,3,3,2,2
250 '
260 '------ Entree de la position initial
e
270 '
280 PRINT#4," Entrez dans les cases 8 ch
iffres"
290 PRINT#4," Barre d'espace pour la cas
e vide"
300 LOCATE 2,7:PRINT "POSITION INITIALE"
310 x0=100:y0=130
320 GOSUB 790:' GRILLE
330 GOSUB 880:'ENTREE
340 WINDOW#3,26,35,10,19:PAPER#3,0:CLS#3
350 WINDOW#3,25,34,9,18:PAPER#3,10:CLS#3
360 LOCATE 22,7:PRINT "POSITION COURANTE
"
370 x0=405:y0=130:GOSUB 790:'GRILLE
380 LOCATE#4,2,2:PRINT#4,"Distance =";TA
B(18)"Chiffre deplace ="
390 '
400 '------ Boucle de recherche
410 '
420 ne=ne+1:GOSUB 590
430 FOR i=1 TO ns:GOSUB 680:in(i)=dm:NEX
T
440 mi=1000
450 FOR i=1 TO ns
460   IF ch(i)=z0 THEN 490
470   IF in(i)<mi THEN mi=in(i):i0=i:GOT
O 490
480   IF in(i)=mi THEN IF RND<0.5 THEN i
0=i
490 NEXT
500 z0=ch(i0):GOSUB 1090
510 LOCATE#4,12,2:PRINT#4,STRING$(3," ")
520 LOCATE#4,12,2:PRINT#4,mi:LOCATE#4,35
,2:PRINT#4,z0
530 IF mi=0 THEN CLS#4:LOCATE#4,2,2:PRIN
T#4,"J'ai gagne en";ne;"coups.":SOUND 1,
478,50:SOUND 1,379,50:SOUND 1,319,100:LO
CATE 1,1:END
540 cf$(0)=cf$(i0)
550 GOTO 420
```

```
560 '
570 '       SUCCESSEURS
580 '
590 z$=cf$(0):j0=INSTR(z$,"9"):c$=LEFT$(
sc$(j0),1):ns=VAL(c$)
600 FOR i=1 TO ns
610   z$=cf$(0):c$=MID$(sc$(j0),i+1,1):j
1=VAL(c$):ch$=MID$(z$,j1,1):ch(i)=VAL(ch
$)
620   MID$(z$,j0,1)=ch$:MID$(z$,j1,1)="9
":cf$(i)=z$
630 NEXT
640 RETURN
650 '
660 '       DISTANCE MANHATTAN
670 '
680 dm=0
690 FOR j=1 TO 9
700   c$=MID$(cf$(i),j,1):ch=VAL(c$):x=(
j-1) MOD 3+1:y=(j-1)\3+1
710   dx=u(ch)-x:dy=v(ch)-y
720   IF ch=1 THEN ft=16 ELSE ft=1
730   dm=dm+ft*(dx*dx*dx*dx+dy*dy*dy*dy)
740 NEXT j
750 RETURN
760 '
770 '       GRILLE
780 '
790 ORIGIN x0,y0:del=40
800 FOR x=0 TO 3
810   PLOT x*del,0:DRAWR 0,3*del
820   PLOT 0,x*del:DRAWR 3*del,0
830 NEXT x
840 RETURN
850 '
860 '       ENTREE
870 '
880 TAG
890 ORIGIN x0+0.3*del,y0+2.7*del
900 FOR y=0 TO 2
910   FOR x=0 TO 2
920     MOVE x*del,-y*del
930     PRINT " ";:'CURSEUR
940     r$=""
950     WHILE r$<>" " AND(r$<"1" OR r$>"
8")
960       r$=INKEY$
970     WEND
980 SOUND 1,100,10
```

```
990       MOVE x*del,-y*del:PRINT r$;
1000        IF r$<>" " THEN cf$(0)=cf$(0)+r
$ ELSE cf$(0)=cf$(0)+"9"
1010    NEXT x
1020 NEXT y
1030 CLS#4
1040 TAGOFF
1050 RETURN
1060 '
1070 '       AFFICHAGE
1080 '
1090 TAG:j=0
1100 ORIGIN x0+0.3*del,y0+2.7*del
1110 FOR y=0 TO 2
1120    FOR x=0 TO 2
1130       MOVE x*del,-y*del
1140       j=j+1:c$=MID$(cf$(i0),j,1):IF c
$="9" THEN PRINT " "; ELSE PRINT c$;
1150    NEXT x
1160 NEXT y
1170 TAGOFF
1180 SOUND 1,100,10
1190 RETURN
```

Essayons le programme

Première tâche : constituer un jeu d'essais, composé d'une liste numérotée de configurations initiales.

5 2	3 8	4 5
1 7 3	2 5 7	7 2 3
4 6 8	1 4 6	1 6 8
CF. 1	*CF. 2*	*CF. 3*
1 4 5	3 7 8	2 5 1
3 6	5 6	6 7
7 8 2	4 1 2	4 8 3
CF. 4	*CF. 5*	*CF. 6*
3 5 1	8 7 6	
2 4	5 4 3	
8 7 6	2 1	
CF. 7	*CF. 8*	

TAQUIN1 a de bonnes chances (le tirage au sort intervient, ne l'oublions pas) de résoudre la première position en 11 coups et la seconde en 15. Mais il peut aussi s'embarquer pour de plus longues randonnées. Pour les autres configurations, les nombres d'essais oscillent entre une trentaine (c'est rare) et une à cinq centaines de coups. Il vous arrivera, sans doute, de désespérer en voyant le programme déplacer les chiffres en rond pendant quelques tours mais il finit généralement par émerger. Heureusement, la succession des essais est relativement rapide.

Rien ne vous empêche d'expérimenter d'autres formules pour calculer la distance et d'en mesurer les résultats. Le sentiment prévaut, cependant, que nous devrions aller plus loin.

L'AMSTRAD JOUE MIEUX

Revenons à notre promenade sur un plateau lunaire en direction de la mer. Nous pourrions, à partir de toute position, envoyer des éclaireurs sur les chemins qui s'ouvrent sans revenir en arrière (soit 1, 2 ou 3 chemins). Un éclaireur avance d'un nombre donné de pas : l'horizon H. Il opère selon une stratégie analogue à celle déjà utilisée. A chaque carrefour, il note les altitudes des divers chemins possibles et choisit celui d'altitude minimale. S'il en est plusieurs, le premier rencontré fera l'affaire. L'information essentielle, recueillie et transmise par l'éclaireur, à son retour, est l'altitude minimale *(minimum minimorum)* rencontrée lors de chaque parcours de H pas. Ces valeurs remplacent, comme critères de choix, les indicateurs du premier niveau associés à chacun des chemins partant de la position actuelle. La recherche de l'altitude minimale est essentielle. Il serait inefficace de se contenter de l'altitude mesurée au terme de chaque parcours.

Nous n'avons pas cru nécessaire d'introduire un tirage au sort, en présence de sentiers d'égale altitude, au cours des cheminements exploratoires. Le hasard a pour rôle principal d'éviter les boucles infinies. Ce risque est ici supprimé par la donnée de l'horizon H. Notre lecteur pourra cependant, s'il le désire, expérimenter cette possibilité.

Le programme TAQUIN2

Pour transformer en TAQUIN2 notre programme TAQUIN1, il suffit donc d'itérer l'usage de notre outillage intellectuel initial. La boucle principale de recherche appellera, à chaque tour, le sous-programme "EXPLORATION" listé ci-dessous. Exception : en cas de deux successeurs, une seule possibilité subsiste, après élimination du retour en arrière.

```
1220 '
1230 '        EXPLORATION
1240 '
1250 FOR k=1 TO ns0
1260    IF ch0(k)=z0 THEN 1400
1270    cf$(0)=cf0$(k):n=0:mimi=1000
1280    GOSUB 610
1290    FOR i=1 TO ns:GOSUB 700:in(i)=dm:
NEXT
1300    mi=1000
1310    FOR i=1 TO ns
1320       IF ch(i)=z THEN 1340
1330       IF in(i)<mi THEN mi=in(i):i0=i
1340    NEXT i
1350    z=ch(i0):mimi=MIN(mimi,mi)
1360    n=n+1:IF n>=h THEN 1390
1370    cf$(0)=cf$(i0)
1380    GOTO 1280
1390    in0(k)=mimi
1400 NEXT k
1410 RETURN
```

Les tables CH(4), CF$(4), IN(4) sont employées, de façon répétitive, par le sous-programme "EXPLORATION". Les valeurs liées à la configuration actuelle sont donc mémorisées en CH0(4), CF0$(4), IN0(4), NS0. La liste complète des variables ajoutées à celles de TAQUIN1 se présente comme suit.

Liste des variables

Tables :

CF0$(4) : variable de stockage pour la position actuelle.
CH0(4) : variable de stockage pour la position actuelle.
IN0(4) : variable de stockage pour la position actuelle.

Variables simples :

H : horizon d'exploration.
K : indice des "éclaireurs".
MIMI : *minimum minimorum.*
N : numéro d'ordre des pas d'exploration.
NS0 : nombre de successeurs de la position initiale.
Z : chiffre déplacé au coup précédent lors de l'exploration.

Les essais comparatifs des deux programmes semblent indiquer qu'à égalité de chances (compte tenu des choix aléatoires dans la boucle principale), le nombre d'essais varie en raison inverse du nombre de

pas (l'horizon H). Voici, à titre de simple résultat d'expériences particulières, quelques valeurs rencontrées pour les configurations d'essai plus haut définis.

Configuration	TAQUIN1	TAQUIN2 (H=4)
1	11	11
2	21	17
3	78	173
4	297	190
5	351	169
6	269	121
7	453	59
8	264	42

Exprimé en temps d'opération, le progrès est beaucoup moins net car chaque phase de calcul dure, en moyenne, trois à quatre secondes avec TAQUIN2 contre moins d'une seconde pour TAQUIN1. TAQUIN2 fournit, par ailleurs, la possibilité d'expérimenter l'incidence sur les résultats, de diverses valeurs de l'horizon H. Notre lecteur pourra s'y essayer.

Listing du programme TAQUIN2

```
10 DEFINT a-z
20 MODE 1:PEN 0:PAPER 13:CLS:PEN#3,0:PEN
#4,0
30 WINDOW#3,17,25,3,5:PAPER#3,0:CLS#3
40 WINDOW#3,16,24,2,4:PAPER#3,10:CLS#3
50 LOCATE#3,2,2:PRINT#3,"TAQUIN2"
60 WINDOW#3,7,16,10,19:PAPER#3,0:CLS#3
70 WINDOW#3,6,15,9,18:PAPER#3,10:CLS#3
80 WINDOW#4,3,39,22,24:PAPER#4,0:CLS#4
90 WINDOW#4,2,38,21,23:PAPER#4,10:CLS#4
100 DIM cf0$(4),cf$(4),sc$(9),in0(4),in(
4),ch0(4),ch(4),u(9),v(9)
110 RANDOMIZE TIME
120 h=4
130 '      Positions suivantes
140 DATA 224,3135,226,3157,42468
150 DATA 3359,248,3579,268
160 FOR i=1 TO 9
170    READ sc$(i)
180 NEXT i
190 '      Solution finale
200 FOR i=1 TO 9
```

```
210   READ u(i),v(i)
220 NEXT i
230 DATA 1,1,2,1,3,1
240 DATA 1,2,3,2,1,3
250 DATA 2,3,3,3,2,2
260 '
270 '------ Entree de la position initial
e
280 '
290 PRINT#4," Entrez dans les cases 8 ch
iffres"
300 PRINT#4," Barre d'espace pour la cas
e vide"
310 LOCATE 2,7:PRINT "POSITION INITIALE"
320 x0=100:y0=130
330 GOSUB 810:' GRILLE
340 GOSUB 900:'ENTREE
350 WINDOW#3,26,35,10,19:PAPER#3,0:CLS#3
360 WINDOW#3,25,34,9,18:PAPER#3,10:CLS#3
370 LOCATE 22,7:PRINT "POSITION COURANTE
"
380 x0=405:y0=130:GOSUB 810:'GRILLE
390 LOCATE#4,2,2:PRINT#4,"Distance =";TA
B(18)"Chiffre deplace ="
400 '
410 '------ Boucle de recherche
420 '
430 ne=ne+1:GOSUB 610
440 FOR i=0 TO ns:ch0(i)=ch(i):cf0$(i)=c
f$(i):NEXT
450 ns0=ns:IF ns0=2 AND z0>0 THEN 460 EL
SE GOSUB 1250
460 mi=1000
470 FOR i=1 TO ns0
480   IF ch0(i)=z0 THEN 510
490   IF in0(i)<mi THEN mi=in0(i):i0=i:G
OTO 510
500   IF in0(i)=mi THEN IF RND<0.5 THEN
i0=i
510 NEXT i
520 z0=ch0(i0):GOSUB 1110
530 cf$(0)=cf0$(i0):i=0:GOSUB 700
540 LOCATE#4,12,2:PRINT#4,STRING$(3," ")
550 LOCATE#4,12,2:PRINT#4,dm:LOCATE#4,35
,2:PRINT#4,z0
560 IF mi=0 THEN CLS#4:PRINT#4," J'ai ga
gne en";ne;"coups":PRINT#4," La suite es
t evidente.":SOUND 1,478,50:SOUND 1,379,
50:SOUND 1,319,100:LOCATE 1,1:END
```

```
570 GOTO 430
580 '
590 '      SUCCESSEURS
600 '
610 z$=cf$(0):j0=INSTR(z$,"9"):c$=LEFT$(
sc$(j0),1):ns=VAL(c$)
620 FOR i=1 TO ns
630   z$=cf$(0):c$=MID$(sc$(j0),i+1,1):j
1=VAL(c$):ch$=MID$(z$,j1,1):ch(i)=VAL(ch
$)
640   MID$(z$,j0,1)=ch$:MID$(z$,j1,1)="9
":cf$(i)=z$
650 NEXT
660 RETURN
670 '
680 '      DISTANCE MANHATTAN
690 '
700 dm=0
710 FOR j=1 TO 9
720   c$=MID$(cf$(i),j,1):ch=VAL(c$):x=(
j-1) MOD 3+1:y=(j-1)\3+1
730   dx=u(ch)-x:dy=v(ch)-y
740   IF ch=1 THEN ft=16 ELSE ft=1
750   dm=dm+ft*(dx*dx*dx*dx+dy*dy*dy*dy)
760 NEXT
770 RETURN
780 '
790 '      GRILLE
800 '
810 ORIGIN x0,y0:del=40
820 FOR x=0 TO 3
830   PLOT x*del,0:DRAWR 0,3*del
840   PLOT 0,x*del:DRAWR 3*del,0
850 NEXT x
860 RETURN
870 '
880 '      ENTREE
890 '
900 TAG
910 ORIGIN x0+0.3*del,y0+2.7*del
920 FOR y=0 TO 2
930   FOR x=0 TO 2
940     MOVE x*del,-y*del
950     PRINT CHR$(143);:'CURSEUR
960     r$=""
970     WHILE r$<>" " AND(r$<"1" OR r$>"
8")
980        r$=INKEY$
990     WEND
```

```
1000 SOUND 1,100,10
1010      MOVE x*del,-y*del:PRINT r$;
1020      IF r$<>" " THEN cf$(0)=cf$(0)+r
$ ELSE cf$(0)=cf$(0)+"9"
1030    NEXT x
1040 NEXT y
1050 TAGOFF
1060 CLS#4
1070 RETURN
1080 '
1090 '      AFFICHAGE
1100 '
1110 TAG:j=0
1120 ORIGIN x0+0.3*del,y0+2.7*del
1130 FOR y=0 TO 2
1140    FOR x=0 TO 2
1150      MOVE x*del,-y*del
1160      j=j+1:c$=MID$(cf0$(i0),j,1):IF
c$="9" THEN PRINT " "; ELSE PRINT c$;
1170    NEXT x
1180 NEXT y
1190 TAGOFF
1200 SOUND 1,100,10
1210 RETURN
1220 '
1230 '      EXPLORATION
1240 '
1250 FOR k=1 TO ns0
1260    IF ch0(k)=z0 THEN 1400
1270    cf$(0)=cf0$(k):n=0:mimi=1000
1280    GOSUB 610
1290    FOR i=1 TO ns:GOSUB 700:in(i)=dm:
NEXT
1300    mi=1000
1310    FOR i=1 TO ns
1320      IF ch(i)=z THEN 1340
1330      IF in(i)<mi THEN mi=in(i):i0=i
1340    NEXT i
1350    z=ch(i0):mimi=MIN(mimi,mi)
1360    n=n+1:IF n>=h THEN 1390
1370    cf$(0)=cf$(i0)
1380    GOTO 1280
1390    in0(k)=mimi
1400 NEXT k
1410 RETURN
```

L'AMSTRAD JOUE MIEUX ENCORE

Une stratégie presque parfaite mettrait en œuvre l'exploration complète des chemins possibles sur un certain horizon (nombre défini de pas). Soit, pour simplifier, un horizon de deux pas. Chacun de nos éclaireurs, au niveau 1, va cheminer sur l'une des voies qui s'ouvrent, sans retour en arrière, à partir de la position actuelle. Après ce premier pas, il explorera, au niveau 2, tous les chemins qui partent de cette nouvelle position et déterminera les altitudes des carrefours où ils aboutissent. Il notera la valeur minimale de **toutes** les altitudes ainsi rencontrées, le long des routes ou au terme des voies. Ce minimum sera associé au chemin (niveau 1) qui fut affecté à l'éclaireur au départ de la position actuelle. Le joueur (l'ordinateur) choisira, sur ces bases, la route associée au minimum de ces minima *(minimum minimorum).*

Dans une forme différente et sous le nom de MINIMAX, cette méthode sera étendue aux jeux stratégiques à deux joueurs, puis détaillée et commentée dans la deuxième partie de cet ouvrage (chapitres 5 à 7). Revers de la médaille : le nombre des configurations ou nœuds explorés croît de façon exponentielle avec l'horizon. Mémoire et temps de calcul en supportent (mal) les conséquences. Il convient donc de limiter le nombre de pas. Nous choisirons ici quatre niveaux.

Il convient, par ailleurs, de gérer au mieux la mémoire utilisée. Pour ce faire, nous retiendrons, **à chaque niveau,** les seules informations indispensables, dont toutes les autres se déduiront, à savoir :

- la configuration courante à ce niveau,
- le nombre de ses successeurs,
- la position de la case vide,
- la dernière valeur calculée du minimum associé à cette configuration.

L'algorithme décrit s'incarne dans le sous-programme EXPLORATION 2 du programme TAQUIN3. Voici, précédé d'une liste de variables complémentaires, le listing de ce programme.

Liste des variables

Tables :

CP(2) : mémorisation d'un chiffre à chaque niveau.

P$(2) : mémorisation des configurations courantes à chaque niveau.

NS(3) : variables mémorisant les nombres de successeurs des configurations courantes à chaque niveau.

J0(3) : variables mémorisant les positions de la case vide pour les configurations courantes à chaque niveau.

Variables simples :

I0 : variable intermédiaire.
I1, I2, I3 : indices de boucles.

Listing du programme TAQUIN3

```
10 DEFINT a-z
20 MODE 1:PEN 0:PAPER 13:CLS:PEN#3,0:PEN
#4,0
30 WINDOW#3,17,25,3,5:PAPER#3,0:CLS#3
40 WINDOW#3,16,24,2,4:PAPER#3,10:CLS#3
50 LOCATE#3,2,2:PRINT#3,"TAQUIN3"
60 WINDOW#3,7,16,10,19:PAPER#3,0:CLS#3
70 WINDOW#3,6,15,9,18:PAPER#3,10:CLS#3
80 WINDOW#4,3,39,22,24:PAPER#4,0:CLS#4
90 WINDOW#4,2,38,21,23:PAPER#4,10:CLS#4
100 DIM cf0$(4),cf$(4),sc$(9),p$(2),cp(2
),in(4),ch0(4),ch(4),u(9),v(9),ns(3),j0(
3)
110 RANDOMIZE TIME
120 '      Positions suivantes
130 DATA 224,3135,226,3157,42468
140 DATA 3359,248,3579,268
150 FOR i=1 TO 9
160    READ sc$(i)
170 NEXT i
180 '      Solution finale
190 FOR i=1 TO 9
200   READ u(i),v(i)
210 NEXT i
220 DATA 1,1,2,1,3,1
230 DATA 1,2,3,2,1,3
240 DATA 2,3,3,3,2,2
250 '
260 '----- Entree de la position initial
e
270 '
280 PRINT#4," Entrez dans les cases 8 ch
iffres"
290 PRINT#4," Barre d'espace pour la cas
e vide"
300 LOCATE 2,7:PRINT "POSITION INITIALE"
310 x0=100:y0=130
320 GOSUB 780:' GRILLE
330 GOSUB 870:'ENTREE
340 LOCATE 22,7:PRINT "POSITION COURANTE
"
```

```
350 WINDOW#3,26,35,10,19:PAPER#3,0:CLS#3
360 WINDOW#3,25,34,9,18:PAPER#3,10:CLS#3
370 x0=405:y0=130:GOSUB 780:'GRILLE
380 LOCATE#4,2,2:PRINT#4,"Distance =";TA
B(18)"Chiffre deplace ="
390 '
400 '------ Boucle de recherche
410 '
420 ne=ne+1:z$=cf$(0):GOSUB 590
430 FOR i=0 TO ns:z$=cf$(0):GOSUB 610:ch
0(i)=ch(i):cf0$(i)=z$:NEXT
440 ns0=ns:GOSUB 1220:mi=1000
450 FOR i=1 TO ns0
460   IF ch0(i)=z0 THEN 490
470   IF in(i)<mi THEN mi=in(i):i0=i:GOT
O 490
480   IF in(i)=mi THEN IF RND<0.5 THEN i
0=i
490 NEXT i
500 z0=ch0(i0):GOSUB 1080
510 z$=cf0$(i0):GOSUB 670
520 LOCATE#4,12,2:PRINT#4,STRING$(3," ")
530 LOCATE#4,12,2:PRINT#4,dm:LOCATE#4,35
,2:PRINT#4,z0
540 IF mi=0 THEN CLS#4:PRINT#4," J'ai ga
gne en";ne;"coups":PRINT#4," La suite es
t evidente.":SOUND 1,478,50:SOUND 1,379,
50:SOUND 1,319,100:LOCATE 1,1:END
550 cf$(0)=z$:GOTO 420
560 '
570 '      SUCCESSEURS
580 '
590 j0=INSTR(z$,"9"):c$=LEFT$(sc$(j0),1)
:ns=VAL(c$)
600 RETURN
610 c$=MID$(sc$(j0),i+1,1):j1=VAL(c$):ch
$=MID$(z$,j1,1):ch(i)=VAL(ch$)
620 MID$(z$,j0,1)=ch$:MID$(z$,j1,1)="9"
630 RETURN
640 '
650 '      DISTANCE MANHATTAN
660 '
670 dm=0
680 FOR j=1 TO 9
690   c$=MID$(z$,j,1):ch=VAL(c$):x=(j-1)
 MOD 3+1:y=(j-1)\3+1
700   dx=u(ch)-x:dy=v(ch)-y
710   IF ch=1 THEN ft=16 ELSE ft=1
720   dm=dm+ft*(dx*dx*dx*dx+dy*dy*dy*dy)
```

```
730 NEXT j
740 RETURN
750 '
760 '      GRILLE
770 '
780 ORIGIN x0,y0:del=40
790 FOR x=0 TO 3
800   PLOT x*del,0:DRAWR 0,3*del
810   PLOT 0,x*del:DRAWR 3*del,0
820 NEXT x
830 RETURN
840 '
850 '      ENTREE
860 '
870 TAG
880 ORIGIN x0+0.3*del,y0+2.7*del
890 FOR y=0 TO 2
900   FOR x=0 TO 2
910     MOVE x*del,-y*del
920     PRINT CHR$(143);:'CURSEUR
930     r$=""
940     WHILE r$<>" " AND(r$<"1" OR r$>"
8")
950       r$=INKEY$
960     WEND
970 SOUND 1,100,10
980     MOVE x*del,-y*del:PRINT r$;
990     IF r$<>" " THEN cf$(0)=cf$(0)+r$
 ELSE cf$(0)=cf$(0)+"9"
1000   NEXT x
1010 NEXT y
1020 TAGOFF
1030 CLS#4
1040 RETURN
1050 '
1060 '      AFFICHAGE
1070 '
1080 TAG:j=0
1090 ORIGIN x0+0.3*del,y0+2.7*del
1100 FOR y=0 TO 2
1110   FOR x=0 TO 2
1120     MOVE x*del,-y*del
1130     j=j+1:c$=MID$(cf0$(i0),j,1):IF
c$="9" THEN PRINT " "; ELSE PRINT c$;
1140   NEXT x
1150 NEXT y
1160 TAGOFF
1170 SOUND 1,100,10
1180 RETURN
```

```
1190 '
1200 '      EXPLORATION 2
1210 '
1220 FOR k=1 TO ns0
1230   IF ch0(k)=z0 THEN 1390
1240   cp(0)=ch0(k):in(k)=1000:z$=cf0$(k
):GOSUB 590:ns(1)=ns:j0(1)=j0
1250   FOR i1=1 TO ns(1)
1260     z$=cf0$(k):i=i1:j0=j0(1):GOSUB
610:p$(1)=z$:GOSUB 670:in(k)=MIN(in(k),d
m)
1270     IF ch(i)=cp(0) THEN 1380 ELSE c
p(1)=ch(i)
1280     GOSUB 590:ns(2)=ns:j0(2)=j0
1290     FOR i2=1 TO ns(2)
1300       z$=p$(1):i=i2:j0=j0(2):GOSUB
610:p$(2)=z$:GOSUB 670:in(k)=MIN(in(k),d
m)
1310       IF ch(i)=cp(1) THEN 1370 ELSE
 cp(2)=ch(i)
1320       GOSUB 590:ns(3)=ns:j0(3)=j0
1330       FOR i3=1 TO ns(3)
1340         z$=p$(2):i=i3:j0=j0(3):GOSU
B 610
1350         IF ch(i)=cp(2) THEN 1360 EL
SE GOSUB 670:in(k)=MIN(in(k),dm)
1360       NEXT i3
1370     NEXT i2
1380   NEXT i1
1390 NEXT k
1400 RETURN
```

Essais de TAQUIN3

Le comportement du programme, vous le verrez, est encourageant. Voici, pour simple information, les résultats relevés au cours d'essais sur nos configurations tests.

Numéro de configuration	*Nombre de coups*
1	9
2	15
3	38
4	150
5	23
6	35
7	151
8	55

RECONNAITRE UN CHIFFRE

Proposons-nous d'enseigner à notre Amstrad les formes graphiques des 10 chiffres (0 à 9) de la numération décimale, afin qu'il sache les reconnaître. Au terme de l'opération, il devra, par exemple, en présence de la figure suivante :

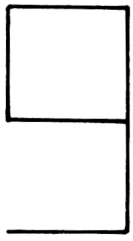


Fig. 1-4

ou d'un codage équivalent, la désigner par un 9.

Pour représenter les chiffres, nous utiliserons le canevas à sept segments que les montres digitales ont popularisé. Les segments seront numérotés comme suit :

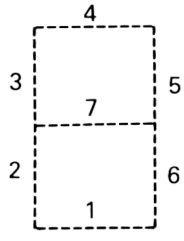


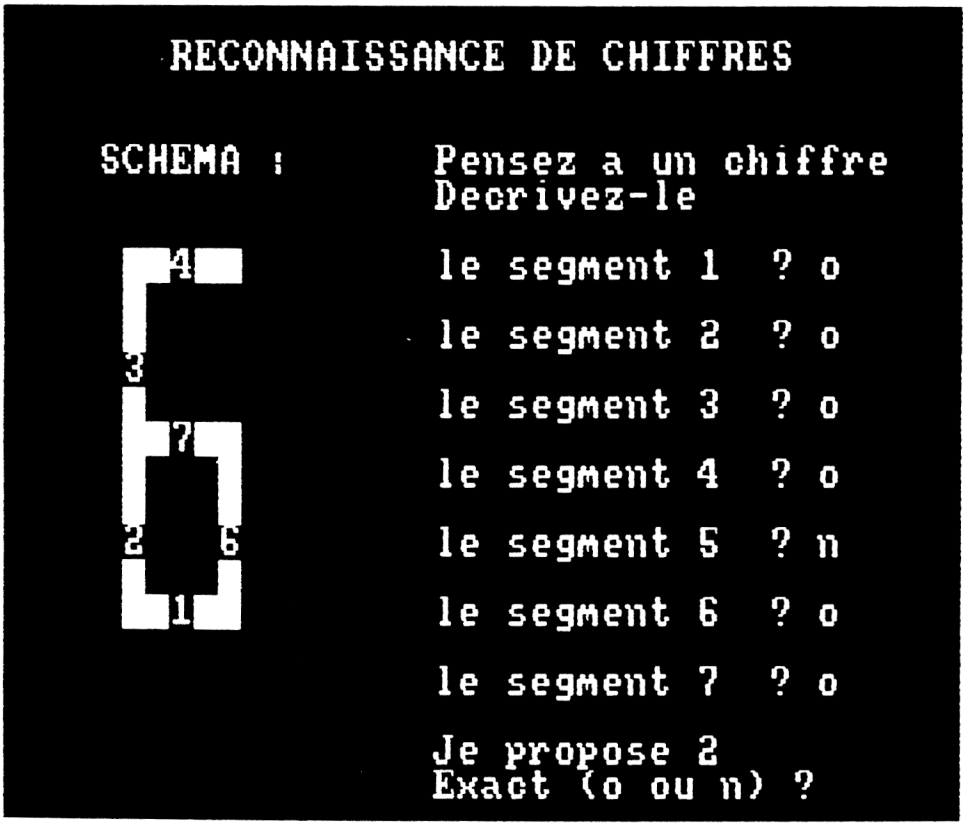
Fig. 1-5

Il serait possible, dans une version évoluée du programme, de dessiner à l'écran un chiffre, selon ce schéma, puis de demander à l'ordinateur de le reconnaître. Au stade actuel, nous pallierons l'absence de vision du micro-ordinateur en lui indiquant, pour chaque chiffre, les numéros des segments qui le forment. De façon précise, nous répondrons par Oui ou Non, pour chaque chiffre évoqué, à une série de 7 questions. Pour le chiffre 6, par exemple, la série des questions et réponses sera :

Contient-il le segment 1 ? Oui
Contient-il le segment 2 ? Oui
Contient-il le segment 3 ? Oui
Contient-il le segment 4 ? Oui
Contient-il le segment 5 ? Non

Contient-il le segment 6 ? Oui
Contient-il le segment 7 ? Oui

Le jeu consiste à penser une série de chiffres puis à décrire chacun d'eux, selon le mode indiqué, à l'intention du micro-ordinateur. Ce dernier, en réponse, devra, chaque fois, deviner le chiffre choisi. Echecs et succès lui sont signalés par Oui ou Non. Le programme "CHIFFRES", nous verrons bientôt comment, apprend par l'expérience. A un moment donné, il domine son sujet et propose à l'opérateur de vérifier qu'il en est bien ainsi. Au cours de l'apprentissage, l'écran ressemble à ceci :



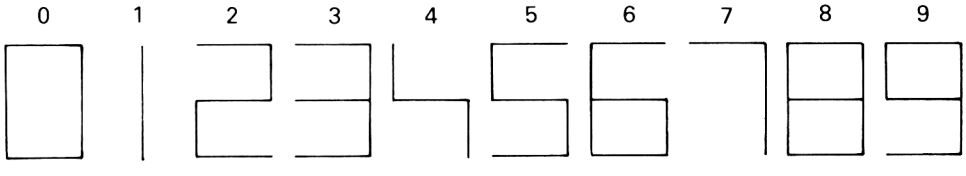
Au terme de l'épreuve il présente l'aspect ci-dessous :

La procédure d'apprentissage

Les éléments de l'espace des solutions comprennent :

- les dix symboles numériques de 0 à 9,
- dix configurations de segments.

La solution réelle du problème est une certaine application d'un ensemble sur l'autre, soit :



L'espace des solutions possibles contient 10 ! (factorielle 10), soit 3 628 800 telles applications. En effet, affecter une figure au symbole 0 peut être réalisé de 10 façons différentes. Restent 9 possibilités pour le

chiffre 1 et ainsi de suite. Essayer de deviner la bonne parmi ces millions d'applications serait une tâche désespérée... et d'ailleurs inutile.

Le problème est, en effet, "séparable" : il peut être décomposé en 10 sous-problèmes, associés à chaque configuration graphique. L'un deux s'énonce par exemple :

Quel chiffre associer à la forme ∃ ?

Chaque sous-problème admet seulement 10 solutions possibles.

Notre première tâche sera de trouver une représentation convenable des formes graphiques à l'intérieur du schéma à 7 segments. Idée simple : représenter par un chiffre binaire (0 ou 1) la réponse Oui ou Non à la question :

La forme évoquée contient-elle le segment no. K ?

Si nous affectons à chaque chiffre binaire un bit de rang K-1, écrit de droite à gauche à partir du rang 0, nous pourrons construire, pour chaque forme graphique, un nombre, en binaire, qui la caractérise. Voici la suite de ces nombres :

Forme :	0	1	2	3
Nombre binaire :	0111111,	0110000,	1011011,	1111001,
Nombre décimal :	63	48	91	121
Forme :	4	5	6	7
Nombre binaire :	1100100,	1101101,	1101111,	0111000
Nombre décimal :	100	109	111	56
Forme :	8	9		
Nombre binaire :	1111111,	1111101		
Nombre décimal :	127	125		

Le nombre décimal z, caractéristique d'une configuration, sera aisément calculé : c'est la somme des puissances de 2 (termes 2^(K-1)) associées à chaque rang K-1, pour tout segment numéroté K à réponse positive. L'opération est réalisée par le sous-programme "QUESTIONS".

```
360 '
370 '       QUESTIONS
380 '
390 CLS#2:CLS #1:z=0
400 PRINT#1,"Contient-il (o/n)"
410 FOR i=1 TO 7
420     PRINT#1
430     PRINT#1,"le segment";i;" ? ";
440     GOSUB 690:'REPONSE OUI/NON
```

```
450     IF r$="o" THEN a(i)=1:GOSUB 780:z
=z+2^(i-1):SOUND 1,35,10,,,1 ELSE SOUND
1,402,10,,,1
460 NEXT i
470 FOR i=1 TO 7:a(i)=0:NEXT
480 PRINT#1
490 RETURN
```

Après transformation de chaque configuration graphique en un nombre caractéristique z, le programme doit apprendre les liaisons entre les 10 nombres z et les chiffres décimaux (de 0 à 9). Pour ce faire, il procède comme toute personne possédant un minimum de mémoire et de bon sens.

A chaque essai, après avoir pris connaissance de la forme graphique présentée, le programme propose un chiffre I. Au départ, celui-ci sera tiré complètement au hasard. Les réponses (Oui en cas de succès, Non en cas d'échec) seront utilisées différemment :

a) si la réponse est Oui, le nombre z calculé deviendra l'élément de rang I d'une table V(I) ;

b) la réponse Non déclenchera le stockage de z dans la première position vide, d'indice I, de trois "tables d'erreur" E1(I), E2(I), E3(I). Si toutes trois contiennent un élément d'indice I non nul, z ira dans E3(I).

A mesure de l'apprentissage, les tables se remplissent et le tirage au sort des propositions du programme est de plus en plus filtré, ce de trois façons :

a) si le nombre caractéristique z a déjà été identifié (il existe un indice I tel que V(I)=z), la proposition est faite à coup sûr (c'est le I !) ;

b) si le chiffre I, tiré au hasard, a été reconnu comme déjà associé à une autre configuration (soit V(I)<>0), un nouveau tirage est réalisé ;

c) il en sera de même si z est l'un des éléments E1(I), E2(I), E3(I) associés au chiffre proposé I. La suggestion a déjà été faite et ne convient pas.

La tactique correspondante est mise en œuvre dans le sous-programme apprentissage, listé ci-dessous.

```
500 '
510 '       APPRENTISSAGE
520 '
530 IF z<>63 AND z<>48 AND z<>91 AND z<>
121 AND z<>100 AND z<>109 AND z<>111 AND
 z<>56 AND z<>127 AND z<>125 THEN PRINT#
1,"Erreur !":FOR j=1 TO 2000:NEXT:RETURN
540 FOR i=0 TO 9
```

```
550     IF v(i)=z THEN PRINT#1,"C'est le"
;i:FOR j=1 TO 2000:NEXT:RETURN
560 NEXT i
570 i=INT(RND*10)
580 IF v(i)<>0 THEN 570
590 IF e1(i)=z OR e2(i)=z OR e3(i)=z THE
N 570
600 ne=ne+1:PRINT#1,"Je propose";i;"Exac
t (o ou n) ? ";
610 GOSUB 690:'REPONSE OUI/NON
620 IF r$="o" THEN v(i)=z:RETURN
630 IF e1(i)=0 THEN e1(i)=z:RETURN
640 IF e2(i)=0 THEN e2(i)=z ELSE e3(i)=z
650 RETURN
```

Le programme CHIFFRES

La structure en est simple :

- préparation de l'écran : titre, messages, mode de représentation des chiffres, ouverture de fenêtres (lignes 10 à 140) ;
- boucle d'apprentissage (lignes 140 à 220) ;
- boucle de vérification des connaissances (ligne 230 à 440) ;
- sous-programmes.

Le nombre d'essais nécessaires pour apprendre la forme des 10 chiffres est intéressant à mesurer. Il varie naturellement à chaque épreuve, compte tenu du caractère aléatoire de la procédure utilisée. La limite théorique optimale avoisine 22 (voir chapitre 4). Les valeurs réelles oscillent entre 25 et 70 essais.

Voici les listes de variables et d'instructions du programme "CHIFFRES".

Listes des variables

Tables :

A$(9)	: chaînes représentatives des segments.
A(9)	: drapeaux pour segments allumés.
E1(10), E2(10), E3(10)	: servent à mémoriser les erreurs.
V(I)	: enregistre les succès.

Variables simples :

D$	: chaîne particulière de caractères.
I,J	: indices de boucles.
L$	: chaîne particulière de caractères.
NE	: nombre d'essais.

P$: chaîne particulière de caractères.
R$: réponse à une question.
T$: chaîne particulière de caractères.
Z : nombre caractéristique d'une forme représentant un chiffre.

Listing du programme "CHIFFRES"

```
10 DEFINT a-z
20 t$=CHR$(143):p$=CHR$(143)+"     ":l$="
    "+CHR$(143):d$=CHR$(143)+"    "+CHR$(
143)
30 RANDOMIZE TIME
40 '
50 ' Organisation de l'ecran
60 '
70 MODE 1:PEN 0:PAPER 13:CLS
80 WINDOW#1,19,39,8,23:PAPER#1,10:PEN#1,
0
90 WINDOW#2,6,10,8,19:PAPER#2,13:PEN#2,0
100 LOCATE 8,2:PRINT "RECONNAISSANCE DE
CHIFFRES"
110 LOCATE 5,5:PRINT "SCHEMA :"
120 LOCATE 19,5:PRINT "Pensez a un chiff
re"
130 LOCATE 19,6:PRINT "Decrivez-le"
140 '
150 '------ Boucle d'apprentissage
160 '
170 ne=0
180 GOSUB 390:'QUESTIONS
190 GOSUB 530:'APPRENTISSAGE
200 FOR i=0 TO 9
210     IF v(i)=0 THEN 180
220 NEXT i
230 '
240 '------ Verification des connaissance
s
250 '
260 CLS#1:PRINT#1:PRINT#1,"J'ai tout com
pris en";ne;"essais !"
270 PRINT#1:PRINT#1,"Verification ? ";
280 r$=LOWER$(INKEY$):IF r$="" THEN 280
290 PRINT#1,r$:IF r$="n" THEN CLS:END
300 GOSUB 390:'QUESTIONS
310 GOSUB 530:'APPRENTISSAGE
320 GOTO 260
```

```
330 '
340 '       Sous-programmes
350 '
360 '
370 '       QUESTIONS
380 '
390 CLS#2:CLS #1:z=0
400 PRINT#1,"Contient-il  (o/n)"
410 FOR i=1 TO 7
420     PRINT#1
430     PRINT#1,"le segment";i;" ? ";
440     GOSUB 690:'REPONSE OUI/NON
450     IF r$="o" THEN a(i)=1:GOSUB 780:z
=z+2^(i-1):SOUND 1,35,10,,,1 ELSE SOUND
1,402,10,,,1
460 NEXT i
470 FOR i=1 TO 7:a(i)=0:NEXT
480 PRINT#1
490 RETURN
500 '
510 '       APPRENTISSAGE
520 '
530 IF z<>63 AND z<>48 AND z<>91 AND z<>
121 AND z<>100 AND z<>109 AND z<>111 AND
 z<>56 AND z<>127 AND z<>125 THEN PRINT#
1,"Erreur !":FOR j=1 TO 2000:NEXT:RETURN
540 FOR i=0 TO 9
550     IF v(i)=z THEN PRINT#1,"C'est le"
;i:FOR j=1 TO 2000:NEXT:RETURN
560 NEXT i
570 i=INT(RND*10)
580 IF v(i)<>0 THEN 570
590 IF e1(i)=z OR e2(i)=z OR e3(i)=z THE
N 570
600 ne=ne+1:PRINT#1,"Je propose";i;"Exac
t (o ou n) ? ";
610 GOSUB 690:'REPONSE OUI/NON
620 IF r$="o" THEN v(i)=z:RETURN
630 IF e1(i)=0 THEN e1(i)=z:RETURN
640 IF e2(i)=0 THEN e2(i)=z ELSE e3(i)=z
650 RETURN
660 '
670 '       REPONSE OUI/NON
680 '
690 r$=""
700 WHILE r$<>"o" AND r$<>"n"
710     r$=LOWER$(INKEY$)
720 WEND
730 PRINT#1,r$
```

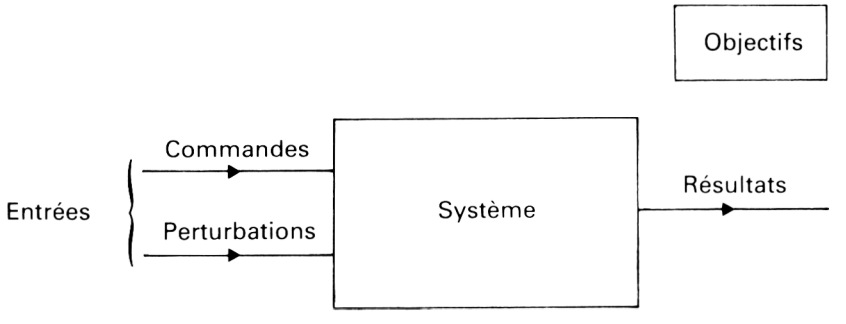
```
740 RETURN
750 '
760 ' AFFICHAGE 7 SEGMENTS
770 '
780 a$(1)=t$+t$+"1"+t$+t$
790 a$(2)=p$+p$+p$+"2     "+p$+p$
800 a$(3)=p$+p$+p$+"3     "+p$+p$
810 a$(4)=t$+t$+"4"+t$+t$
820 a$(5)=l$+l$+l$+"     5"+l$+l$
830 a$(6)=l$+l$+l$+"     6"+l$+l$
840 a$(7)=t$+t$+"7"+t$+t$
850 a$(8)=d$+d$+d$+"3    5"+d$+d$
860 a$(9)=d$+d$+d$+"2    6"+d$+d$
870 CLS#2
880 IF a(1)=1 THEN LOCATE#2,1,11:PRINT#2
,a$(1)
890 IF a(2)=1 THEN IF a(1)=1 THEN a$(2)=
LEFT$(a$(2),25):LOCATE#2,1,6:PRINT#2,a$(
2) ELSE LOCATE#2,1,6:PRINT#2,a$(2)
900 IF a(3)=1 THEN LOCATE#2,1,1:PRINT#2,
a$(3)
910 IF a(4)=1 THEN LOCATE#2,1,1:PRINT#2,
a$(4)
920 IF a(5)=1 THEN IF (a(3)=1) OR (a(4)=
1) THEN 930 ELSE LOCATE#2,1,1:PRINT#2,a$
(5):GOTO 950
930 IF (a(5)=1) AND (a(4)=1) THEN IF a(3
)<>1 THEN a$(5)=RIGHT$(a$(5),25):LOCATE#
2,1,2:PRINT#2,a$(5) ELSE a$(8)=RIGHT$(a$
(8),25):LOCATE#2,1,2:PRINT#2,a$(8):GOTO
950
940 IF (a(5)=1) AND (a(3)=1) THEN LOCATE
#2,1,1:PRINT#2,a$(8)
950 IF a(6)=1 THEN IF (a(1)=1) OR (a(2)=
1) THEN 960 ELSE LOCATE#2,1,6:PRINT#2,a$
(6):GOTO 980
960 IF (a(6)=1) AND (a(1)=1) THEN IF a(2
)<>1 THEN a$(6)=LEFT$(a$(6),25):LOCATE#2
,1,6:PRINT#2,a$(6) ELSE a$(9)=LEFT$(a$(9
),25):LOCATE#2,1,6:PRINT#2,a$(9):GOTO 98
0
970 IF (a(6)=1) AND (a(2)=1) THEN LOCATE
#2,1,6:PRINT#2,a$(9)
980 IF a(7) THEN LOCATE#2,1,6:PRINT#2,a$
(7)
990 RETURN
```

Chapitre 2
UNE IDÉE PUISSANTE : LA RÉTROACTION

SYSTÈMES AUTOCONTROLÉS

Revenons, un instant, à la définition générale d'un problème, construite au début du chapitre 1. Elle distinguait un "générateur" de solutions possibles et un "vérificateur" des conditions posées. Les difficultés, avons-nous souligné, concernent les instructions à donner au générateur pour une exploration efficace de l'espace des solutions. Une connaissance un peu précise de la *structure* du problème peut guider cette recherche. Nous allons le voir sur un cas particulier important.

Pour nous rapprocher des situations concrètes, considérons un système défini par des entrées et des sorties.



Les entrées comprennent des commandes, sous le contrôle du générateur de solutions et des perturbations : incidences du milieu extérieur et modifications des paramètres de structure du système. Les sorties sont des résultats ou indicateurs ou encore variables essentielles, selon le langage utilisé.

Soit d'abord un objectif unique. La condition posée, qui définit le problème à résoudre, est l'égalité du résultat et de cet objectif, constant ou variable au cours du temps.

Enfin, et ces points sont essentiels, supposons que :

- une commande au moins prenne des valeurs scalaires (qui puissent être ordonnées). C'est, en particulier, le cas des nombres entiers ou réels ;
- le résultat, également de type scalaire, puisse être modifié, et dans un sens *connu* par l'une au moins des commandes scalaires.

Supposons, par exemple, que le résultat R croisse si une certaine commande C est positive (et inversement). Il suffira alors de lier cette commande à *l'écart* E=0−R (objectif moins résultat), par exemple sous la forme (avec K>0) :

C=K*E=K*(0−R)

Toute valeur positive de l'écart (Résultat < Objectif) donnera à C une valeur positive qui augmentera R, réduira l'écart et donc rapprochera objectif et résultat.

Ce faisant, nous introduisons une *rétroaction* ou *feedback*. Elle associe les opérations suivantes :

a) comparaison, à divers instants, du résultat à l'objectif ;

b) action directe de cet écart sur une commande, dans le sens connu qui, par modification du résultat, réduit l'écart.

Cette simple transformation se révèle d'une extraordinaire puissance : *nous avons créé un système autocontrôlé qui poursuit de lui-même son objectif.*

La méthode est d'application générale. Elle tire parti d'une information limitée mais précise quant à la structure du système : liaison de sens connu entre commande et résultat. A cela près, elle n'exige aucune autre connaissance des processus internes ou des perturbations extérieures. En d'autres termes, nous pouvons ignorer les moyens utilisés et les conditions d'emploi. Un système autocontrôlé est presque entièrement défini par ses objectifs. Des exemples et des programmes nous aideront à le vérifier.

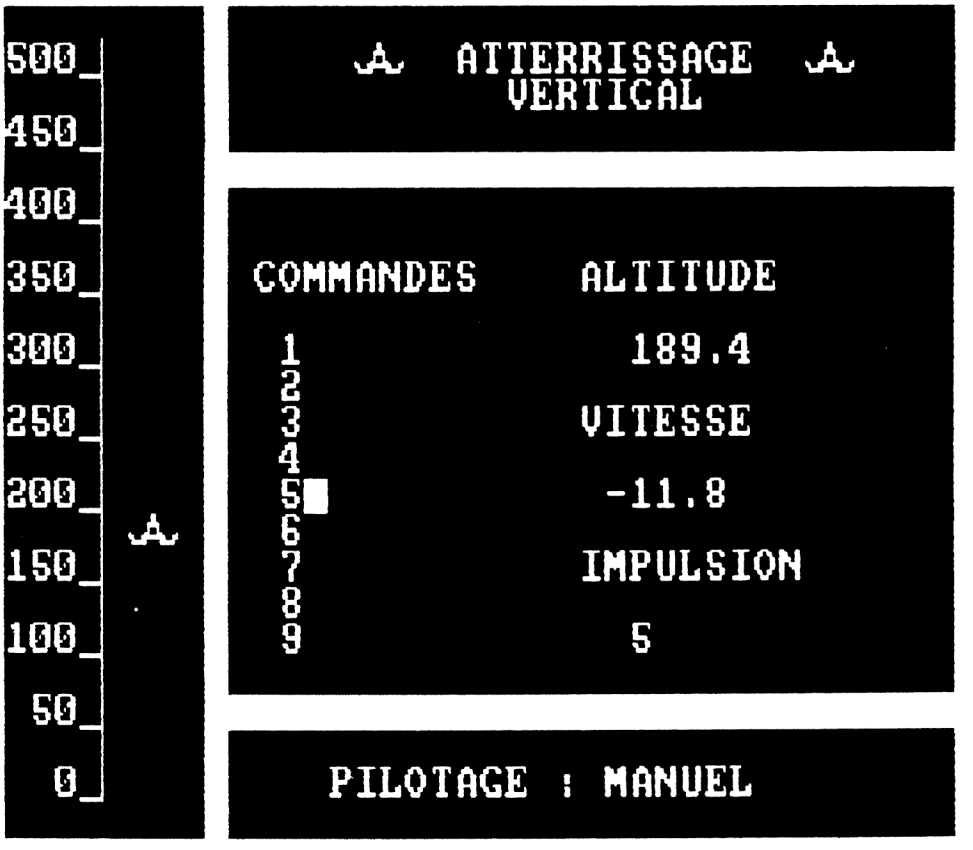
Un pilote automatique

Vous êtes aux commandes d'un engin terrestre capable d'atterrissage vertical. Altitude et vitesse verticale initiales sont définies. La manette des gaz, à neuf positions en pilotage manuel, contrôle un réacteur à poussée verticale. Vous pouvez, à tout moment, enclencher le pilote automatique.

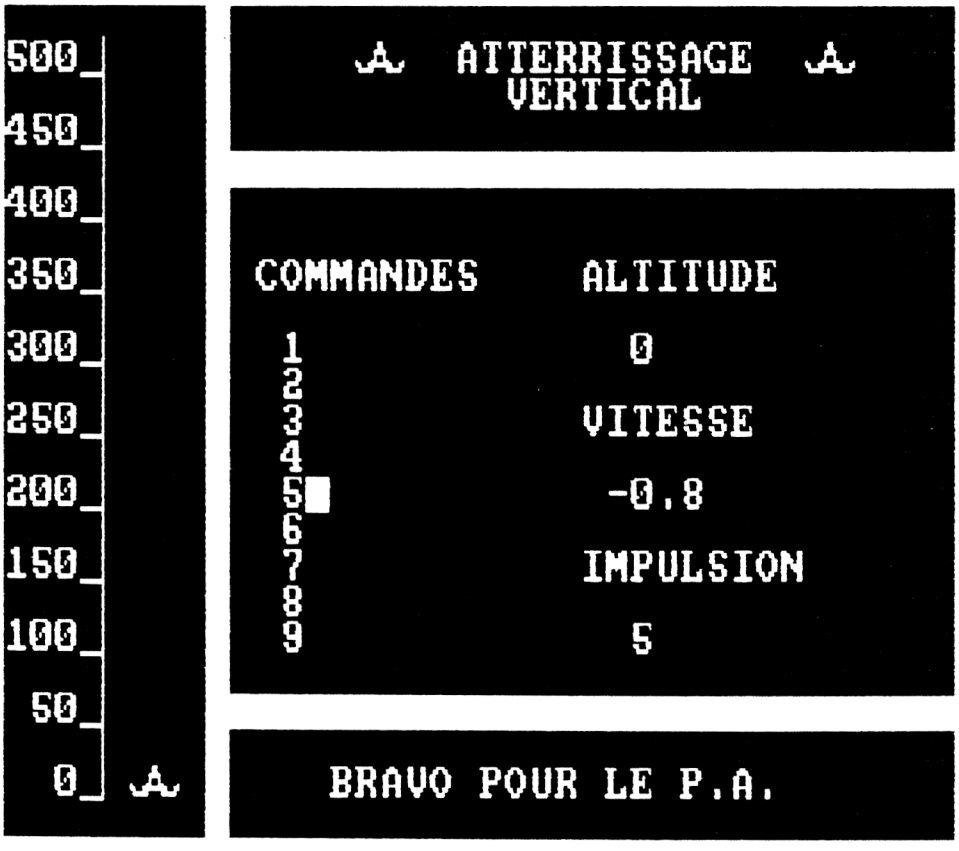
Les manœuvres, notez-le bien, s'effectuent *en temps réel* ; l'engin continue sa course, à partir de la dernière commande reçue, sans attendre la commande suivante.

Objectif : se poser sans casse (vitesse de contact inférieure à 1 mètre/seconde en valeur absolue).

Au cours des opérations, l'écran simule un tableau de bord et offre l'aspect suivant, tant que le pilote automatique n'a pas été mis en service :



Après un atterrissage réussi, avec l'aide du pilote automatique, apparaît ceci :



Les équations du mouvement de l'engin (lois de la Mécanique) sont toutes écrites dans la ligne n° 390 du programme PAT (pilote automatique d'atterrissage) :

```
390 H=H+V:P=K*F:AC=(P-M*G)/M:V=V+AC
```

Notations :

M =masse de l'engin =2.
G =accélération de la pesanteur terrestre =9,8.
F =position de la manette (1 à 9).
P =poussée du réacteur.
K =coefficient de transmission =3,92.
AC =accélération.
V =vitesse.
H =altitude.

L'unité de temps est prise égale au cycle de calcul, fixé à 1 seconde. AC,V,H sont mesurées sur un axe vertical orienté vers le haut. La valeur de K a été calculée afin d'obtenir un vol stationnaire pour la position moyenne (F=5) de la manette.

Un sous-programme, listé ci-dessous, simule le pilote automatique. Celui-ci régule la vitesse autour de deux valeurs négatives (vitesses de descente) :

– 5 m/s pour une altitude supérieure à 20 mètres,
– 0,8 m/s pour une altitude inférieure à 20 mètres.

```
700 '
710 '       PILOTE AUTOMATIQUE
720 '
730 IF h>20 THEN f=5-k1*(v+5) ELSE f=5-k
1*(v+0.8)
740 IF f<1 THEN f=1
750 IF f>9 THEN f=9
760 RETURN
```

La position F de la manette et donc la poussée du réacteur sont liées à l'écart de vitesse par les relations de feedback :

F=5−K1*(V+5) pour H>20
F=5−K1*(V+0,8) pour H<20

Elles font intervenir un coefficient K1 qui règle l'amplification dans la boucle de réaction. La valeur choisie K1=0,3 donne une régulation rapide et stable. Il est passionnant et instructif de faire varier K1 entre 0 et 1 et d'observer les phénomènes d'oscillation qui peuvent en résulter. Dans la réalité, ils mettraient à rude épreuve la résistance de l'engin et de ses occupants.

Le programme PAT

Vous pouvez entrer le programme dont la liste suit, exercer vos réflexes avec ou sans pilote automatique et modifier tous les paramètres qui ne vous conviendront pas. Vous serez peut-être choqué par un déni de justice patent. Les utilisateurs humains du programme PAT n'ont droit, pour prouver leur adresse, qu'à des valeurs entières de la variable F (impulsion). Le pilote automatique, lui, utilise sans vergogne des valeurs décimales. Pour faire régner le bon droit, il suffit, dans le sous-programme "Pilote automatique" (lignes 730 à 760), d'ajouter les instructions :

F=CINT(F)

Ceci fait, la nécessité apparaîtra sans doute d'ajuster, autour de 0,7, la valeur du coefficient K1 de couplage.

Liste des variables

AC : accélération.
F : position de la manette.
G : accélération de la pesanteur.
H : hauteur au-dessus du sol.
HT : hauteur au-dessus du sol pour le graphisme.
I : indice de boucle.
K : coefficient de transmission.
K1 : constante du pilote automatique.
M : masse de l'engin.
M$: chaîne contenant le dessin de l'engin.
P : poussée du réacteur.
R$: caractère lu.
T : instant dans la boucle de calcul.
Z$: chaîne de 6 espaces.

Listing du programme PAT

```
10 '
20 ' ORGANISATION DE L'ECRAN
30 '
40 SYMBOL AFTER 249
50 SYMBOL 250,1,1,3,2,134,207,124,0
60 SYMBOL 251,128,128,192,64,97,243,62,0
70 m$=CHR$(250)+CHR$(251)
80 MODE 1:PAPER 13:PEN 0:CLS
90 WINDOW#3,2,9,2,24:PAPER#3,10:CLS#3:PE
N#3,0
100 WINDOW#5,11,39,7,20:PAPER#5,10:CLS#5
:PEN#5,0
110 WINDOW#2,11,39,2,5:PAPER#2,10:CLS#2:
PEN#2,0
120 LOCATE#2,6,2:PRINT#2,m$;"  ATTERRISS
AGE  ";m$
130 LOCATE#2,12,3:PRINT#2,"VERTICAL"
140 WINDOW#2,11,39,22,24:PAPER#2,10:CLS#
2
150 '
160 ' INITIALISATION
170 '
```

```
180 m=2:g=9.8:k=3.92:f=5:k1=0.3:z$=STRIN
G$(6," "):ht=1
190 PRINT#5,"ENTREZ LES CONDITIONS","INI
TIALES:"
200 PRINT#5:INPUT#5,"ALTITUDE INITIALE (
EN METRES)";h
210 PRINT#5:INPUT#5,"VITESSE INIT.(EN ME
TRES/SEC.)";V
220 LOCATE#5,1,10:PRINT#5,"ETES VOUS PRE
T ? ";:GOSUB 800:'Reponse par oui ou non
230 IF r$="n" THEN LOCATE#5,1,10:PRINT#5
, STRING$(20," "):GOTO 220 ELSE CLS#5
240 LOCATE#5,2,3:PRINT#5,"COMMANDES"
250 FOR i=1 TO 9
260 LOCATE#5,2,4+i:PRINT#5,i
270 NEXT
280 LOCATE#2,5,2:PRINT#2,"PILOTAGE : MAN
UEL"
290 FOR i=1 TO 11
300 LOCATE#3,1,2*i:PRINT#3,USING "###";5
50-50*i;:PRINT#3,"_"
310 NEXT
320 PLOT 78,32
330 FOR i=1 TO 336
340 DRAW 78,32+i,0
350 NEXT
360 ' BOUCLE DE COMMANDE
370 t=TIME:IF r$="a" THEN GOSUB 730 ELSE
 GOSUB 890:'Commande automatique ou manu
elle
380 LOCATE#5,15,3:PRINT#5,"ALTITUDE":LOC
ATE#5,15,7:PRINT#5,"VITESSE":LOCATE#5,15
,11:PRINT#5,"IMPULSION"
390 h=h+v:p=k*f:ac=(p-m*g)/m:v=v+ac
400 SOUND 1,0,100,15,,,31-f
410 IF h<0 THEN h=0
420 FOR i=5 TO 13 STEP 4
430 LOCATE#5,18,i:PRINT#5,z$
440 NEXT
450 LOCATE#3,6,ht:PRINT#3,"   "
460 ht=1:IF h>525 THEN 520
470 FOR i=1 TO 20
480 ht=ht+(h>525-25*i)
490 NEXT
500 ht=21+ht
510 LOCATE#3,6,ht:PRINT#3,m$
520 LOCATE#5,16,5:PRINT#5,ROUND(h,1)
530 LOCATE#5,16,9:PRINT#5,ROUND(v,1)
540 LOCATE#5,16,13:PRINT#5,ROUND(f,1)
```

```
550 IF (TIME-t)<300 THEN 550:'Duree de b
oucle=1 sec
560 IF h>0 THEN 370
570 ' FIN DES OPERATIONS
580 LOCATE 10,15
590 IF v<-1 OR v>1 THEN 610
600 IF VAL(r$)<>0 THEN CLS#2:LOCATE#2,5,
2:PRINT#2,"FELICITATIONS":LOCATE 1,1:END
 ELSE CLS#2:LOCATE#2,5,2:PRINT#2,"BRAVO
POUR LE P.A.":LOCATE 1,1:END
610 FOR i=50 TO 180:SOUND 1,i,1,15:NEXT
620 SOUND 1,0,300,15,1,,16
630 FOR i=1 TO 10
640 PAPER 0:CLS:PAPER 13:CLS
650 NEXT
660 END
670 '
680 '      SOUS-PROGRAMMES
690 '
700 '
710 '      PILOTE AUTOMATIQUE
720 '
730 IF h>20 THEN f=5-k1*(v+5) ELSE f=5-k
1*(v+0.8)
740 IF f<1 THEN f=1
750 IF f>9 THEN f=9
760 RETURN
770 '
780 '      REPONSE PAR OUI OU NON
790 '
800 r$=""
810 WHILE r$<>"o" AND r$<>"n"
820   r$=LOWER$(INKEY$)
830 WEND
840 PRINT#5,UPPER$(r$)
850 RETURN
860 '
870 '       COMMANDE MANUELLE
880 '
890 r$=LOWER$(INKEY$)
900 IF VAL(r$)>=1 AND VAL(r$)<=9 THEN LO
CATE#5,4,4+f:PRINT#5," ":f=VAL(r$):LOCAT
E#5,4,4+f:PRINT#5,CHR$(143):RETURN
910 IF r$="a" THEN LOCATE#2,16,2:PRINT#2
,STRING$(6," "):LOCATE#2,16,2:PRINT#2,"A
UTOMATIQUE":GOSUB 730:'Pilote automatiqu
e
920 RETURN
```

Objectifs multiples

Pour autoriser des rétroactions efficaces, la structure du système doit présenter les caractères suivants :

- résultats et objectifs sont de type scalaire ;
- il est possible de choisir, dans le système considéré, plusieurs entrées, également scalaires, en nombre au moins égal à celui des résultats ;
- chacun des résultats peut être modifié et, dans un sens connu, par l'une au moins des entrées, toutes choses égales d'ailleurs.

Si nous connaissons seulement le sens de l'action d'une commande C sur un résultat R, nous construirons le réseau de contrôle de sorte que l'écart entre R et l'objectif qui lui est lié agisse sur C dans le sens convenable.

Nous disposerons d'une plus grande liberté de synthèse si une même commande C peut agir, dans une direction définie, sur plusieurs résultats R1, R2... Nous pourrons alors modifier l'architecture du réseau et lier, de diverses façons, les écarts aux commandes. La matrice de couplage est choisie en fonction de considérations plus techniques qui n'ont pas leur place ici.

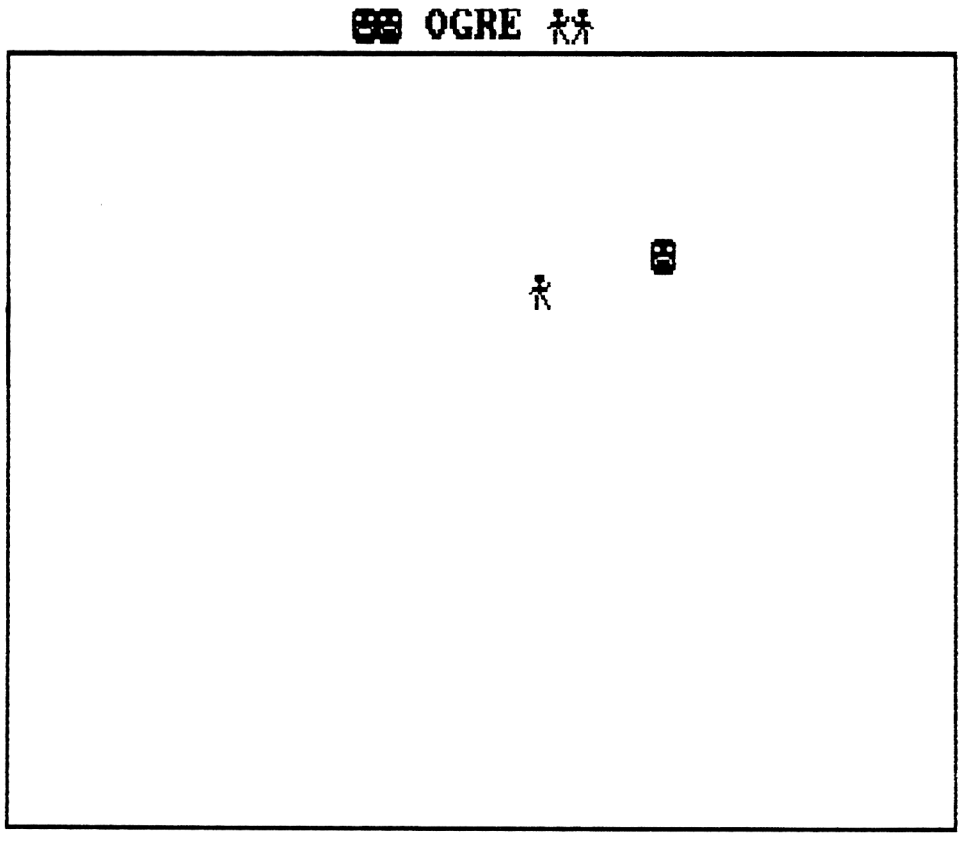
Illustrons ceci par un programme qui pourra également nous distraire.

Le programme OGRE

Il est très simple et met en jeu :

- un garçonnet fuyant, en tous sens, sur l'écran ;
- un ogre qui veut en faire son dîner.

L'ogre, d'esprit moderne, s'est équipé d'engins à tête chercheuse. Celui que nous verrons à l'œuvre détecte la direction et la distance de son objectif. Le signal correspondant commande des moteurs qui lancent, à tout instant, l'engin vers sa cible avec une force proportionnelle à la distance engin-cible. Le programme OGRE nous permet d'assister à cette lutte inégale et d'apprécier les orbes décrites par l'engin. La copie d'écran, statique, n'en peut donner qu'une très faible idée. Le drame se joue jusqu'au moment, aléatoire mais inéluctable, de la capture (signal musical adéquat !).



L'engin a une masse, donc une inertie et obéit aux lois de la Mécanique qu'il convient d'écrire.

Notations :

Système de coordonnées : celui des fenêtres de l'Amstrad (numéros de lignes et de colonnes).

0X, 0Y : coordonnées de la cible.
MX, MY : coordonnées de l'engin.
VX, VY : composantes de vitesse de l'engin.
AX, AY : composantes de l'accélération de l'engin.
FX, FY : composantes de la force agissant sur l'engin.
M : masse de l'engin.
K : coefficient de rétroaction.

Equations :

```
180 fx=k*(ox-mx):fy=k*(oy-my)
190 ax=fx/m:ay=fy/m
```

```
200 vx=vx+ax:vy=vy+ay
210 mx=mx+vx:my=my+vy
```

Listes de variables et d'instructions du programme OGRE

Variables :

Ajouter aux notations précédentes :

L,LI : numéros de lignes.
C, C0 : numéros de colonne.
H : pour alterner les caractères de codes 250 et 251.

Liste d'instructions :

```
10 h=0:RANDOMIZE TIME:INK 0,1:INK 1,24:P
APER 0:PEN 1
20 MODE 1:TAG
30 FOR i=1 TO 110
40 MOVE 2*i,390:PRINT " ";CHR$(224);CHR$
(225);" OGRE ";CHR$(250);CHR$(251);
50 NEXT
60 TAGOFF
70 PLOT 10,20,1
80 DRAW 630,20:DRAW 630,370:DRAW 10,370:
DRAW 10,20
90 ox=20:oy=10:mx=1:my=1:k=2:m=20
100 l=oy:c=ox
110 IF RND<0.5 THEN ox=ox-1 ELSE ox=ox+1
120 IF RND<0.5 THEN oy=oy-1 ELSE oy=oy+1
130 IF (ox<2) OR (ox>39) THEN ox=2-37*(o
x>39)
140 IF (oy<3) OR (oy>23) THEN oy=3-20*(o
y>23)
150 LOCATE c,l:PRINT " "
160 li=oy:co=ox:LOCATE co,li:PEN 10:PRIN
T CHR$(250+h):h=ABS(h-1):PEN 3
170 l=my:c=mx
180 fx=k*(ox-mx):fy=k*(oy-my)
190 ax=fx/m:ay=fy/m
200 vx=vx+ax:vy=vy+ay
210 mx=mx+vx:my=my+vy
220 IF (mx<2) OR (mx>39) THEN mx=2-37*(m
x>39)
230 IF (my<3) OR (my>23) THEN my=3-20*(m
y>23)
240 LOCATE c,l:PRINT " "
```

```
250 li=my:co=mx:LOCATE co,li:PRINT CHR$(
225)
260 IF ROUND(mx)=ox AND ROUND(my)=oy THE
N 280
270 SOUND 1,1000,1,15:GOTO 100
280 ENT 2,=71,20,=119,20,=132,20,=68,0
290 FOR i=1 TO 2:SOUND 1,0,50,15,,2:NEXT
300 LOCATE co,li:PEN 3:PRINT CHR$(224)
310 FOR i=1 TO 2000:NEXT
320 SOUND 1,2000,9,15:
330 LOCATE 1,1:END
```

Ce programme, me direz-vous, semble destiné surtout aux esprits contemplatifs. Pas tout à fait. Rien ne vous empêche de modifier le coefficient K de rappel, non plus que la masse, ou de changer la loi définissant la force agissant sur l'engin :

- force constante ;
- inversement proportionnelle à la distance ;
- inversement proportionnelle au carré de la distance ;
- etc.

et d'en étudier les effets.

STABILITÉ ET PRÉCISION

Il ne suffit pas, dans la pratique, d'associer, dans le sens convenable, un écart à une commande pour obtenir le résultat désiré. Le coefficient de contrôle et la fréquence de mesure des écarts doivent se tenir entre certaines limites.

Le coefficient de contrôle est, nous l'avons vu, le rapport entre la valeur d'une commande et l'écart mesuré entre objectif et situation réelle. Un coefficient trop faible ne permet pas de suivre, avec une précision et une vitesse suffisantes, les évolutions de l'objectif et des perturbations. Par contre, au-delà d'un certain seuil, le système est **instable**. Il devient le siège d'oscillations de grande amplitude qui le rendent inutilisable.

Quant aux écarts, ils ne peuvent être, dans les conditions habituelles, constamment repérés, sous peine de coûts élevés. Si d'ailleurs, comme pour notre Amstrad, le système utilisé est digital, la boucle de calcul a une durée, très faible peut-être, mais non nulle. Il faut donc définir l'intervalle de temps optimal qui séparera deux mesures successives du résultat. La régulation sera, en général, d'autant plus précise que ces mesures seront plus fréquentes. Cependant, il n'est pas efficace d'accroître cette fréquence si, dans l'intervalle entre deux mesures, la situation observée n'évolue pas sensiblement.

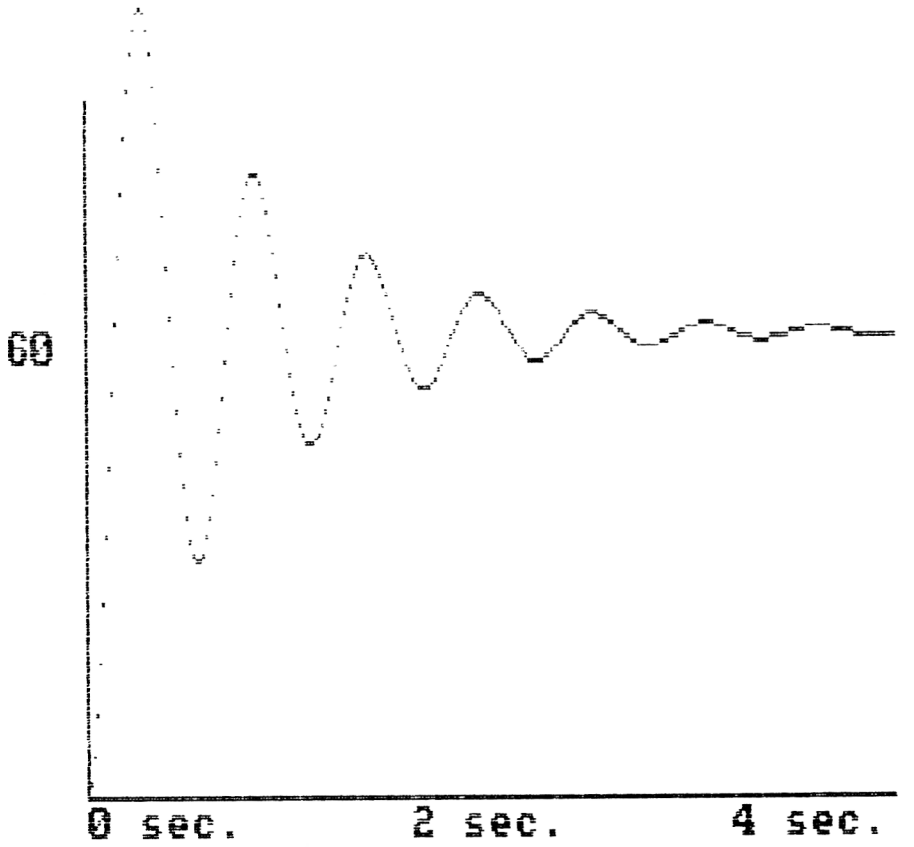
Direction assistée d'un navire

Pour vérifier tout cela, nous allons simuler la commande du gouvernail d'un navire grâce à la barre tenue par le pilote. Cet exemple a valeur de symbole. La "cybernétique", science du contrôle et de la communication, tire en effet son nom du mot grec *kubernetes* qui désignait le pilote d'un navire.

Le système étudié comprend :

- la barre ;
- les capteurs des positions angulaires de cette barre et du gouvernail ;
- un servomoteur commandé par l'écart angulaire ;
- le gouvernail avec sa masse et son inertie ;
- le frottement de l'eau qui s'oppose au mouvement du gouvernail avec une force croissant avec la vitesse angulaire de déplacement.

Soumettons l'ensemble à un test. Partons de positions neutres (angles nuls) du gouvernail et de la barre puis demandons au pilote, vous en l'occurrence, de tourner la barre d'un angle choisi. Le programme CYBER va simuler les mouvements du système, enregistrer dans une table les positions successives du gouvernail et en tracer la courbe sur l'écran de votre Amstrad.



Essayez. Voici les résultats pour une rotation de 60 degrés et un jeu de constantes physiques qui imposent au gouvernail de nombreuses oscillations, d'amplitude décroissante, autour de sa position finale :

– moment d'inertie : 0,5 ;
– coefficient de frottement : 1 ;
– coefficient de contrôle : 40.

Le programme CYBER

Il comprend trois parties :

a) messages explicatifs et entrée des données (moment d'inertie du gouvernail, coefficient de frottement, coefficient de contrôle, position de la barre en degrés) ;

b) sous-programme CALCUL ;

c) sous-programme TRACE DE COURBE.

La première partie n'appelle pas de commentaires. Voici la liste des instructions de la seconde :

```
330 '
340 '          CALCUL
350 '
360 n=0:ang=0:va=0
370 mom=k*(b-ang)-f*va
380 aca=mom/in:va=va+aca*dt:ang=ang+va*d
t
390 n=n+1:ang(n)=ang
400 IF n<nm THEN 370
410 RETURN
```

Variables du sous-programme CALCUL

ACA	: accélération angulaire.
ANG(NM)	: table des positions angulaires du gouvernail au cours du temps.
ANG	: position angulaire courante.
B	: position de la barre (en radians).
DT	: intervalle de temps entre deux calculs.
F	: coefficient de frottement.
IN	: moment d'inertie du gouvernail.
K	: coefficient de contrôle.
MOM	: couple ou moment global exercé sur le gouvernail.
N	: compteur des positions calculées.
NM	: nombre maximal de positions.
VA	: vitesse angulaire du gouvernail.

L'instruction de la ligne 330 exprime que le couple ou moment que supporte le gouvernail est la somme de deux termes :

- le couple du servomoteur, proportionnel à l'écart angulaire instantané entre barre et gouvernail ;
- le couple −F*VA exercé par l'eau, proportionnel à la vitesse angulaire et de sens inverse.

Les instructions de la ligne 340 calculent l'accélération angulaire comme quotient du couple total par le moment d'inertie du gouvernail. A partir de l'accélération, sont évaluées vitesse et position angulaire du gouvernail par intégration point à point. Les positions angulaires sont mémorisées dans la table ANG(NM) jusqu'à la limite NM.

Le tracé s'effectue grâce à un sous-programme standard (*BASIC Amstrad CPC 464* par J. Boisgontier et B. Cesard, Editions du PSI, page 137) convenablement adapté et listé ci-dessous. Il fait appel aux variables suivantes.

Liste des variables

B	: position de la barre en radians.
BA	: position de la barre en degrés.
B1, B2	: bornes de l'axe OX.
DT	: intervalle élémentaire de temps.
EX, EY	: échelles des axes.
HECR	: hauteur utilisée de l'écran en pixels.
I	: indice de boucle.
IECR	: largeur de l'écran en pixels.
SX,SY	: coordonnées d'un point en pixels.
X,Y	: variables intermédiaires.
Y1,Y2	: bornes de l'axe OY.
Z$	: variable de travail.

Listing du sous-programme COURBE

```
420 '
430 '          COURBE
440 '
450 ORIGIN 50,20
460 lecr=500:hecr=300:b1=0:b2=5
470 y1=0:y2=1.5
480 ex=lecr/(b2-b1):ey=hecr/(y2-b1)
490 '--------Trace des axes
500 x=b1:PLOT x,1:DRAW x,hecr,1
510 y=y1:PLOT 1,0:DRAW lecr,y
520 '--------Graduation de l'axe OX
530 TAG
```

```
540 FOR i=0 TO 5 STEP 2
550   x=100*i:y=-5
560   PLOT x,y:z$=MID$(STR$(i),2):PRINT
z$;" sec.";
570 NEXT i
580 '--------Graduation de l'axe OY
590 PLOT -50,200:PRINT"    ";
600 PLOT -50,200:z$=MID$(STR$(ba),2):PRI
NT z$;
610 TAGOFF
620 '--------Trace de la courbe
630 FOR x=b1 TO b2 STEP dt
640   y=ang(x/dt)/b
650   sx=(x-b1)*ex:sy=(y-y1)*ey
660   PLOT sx,sy,1
670 NEXT x
680 RETURN
```

La courbe représentée donne le rapport entre la position angulaire du gouvernail et celle définie pour la barre lors de la commande. Cette dernière est donc prise comme unité.

Listing du programme CYBER

```
10 INK 0,1:INK 1,24:PAPER 0:PEN 1:MODE 1
:TAG
20 FOR i=1 TO 100
30 MOVE 2*i,390:PRINT " CYBER";
40 NEXT
50 TAGOFF
60 dt=0.02:nm=CINT(5/dt):DIM ang(nm)
70 LOCATE 1,5
80 PRINT "  Ce programme simule la direc
tion"
90 PRINT " assistee d'un navire."
100 PRINT "  Il trace la courbe des posi
tions"
110 PRINT " du gouvernail, en reponse"
120 PRINT " a une commande de la barre"
130 PRINT " que vous fixez."
140 PRINT:PRINT "  Entrez, au prealable,
"
150 PRINT " les caracteristiques du syst
eme."
160 LOCATE 1,15:PRINT STRING$(50," ")
170 LOCATE 1,15:INPUT" Moment d'inertie
(0.5 a 2) ";in
```

```
180 IF (in<0.5) OR (in>2) THEN SOUND 1,1
00,30:GOTO 160
190 LOCATE 1,17:PRINT STRING$(50," ")
200 LOCATE 1,17:INPUT" Coefficient de fr
ottement (1 a 5) ";f
210 IF (f<1) OR (f>5) THEN SOUND 1,100,3
0:GOTO 190
220 LOCATE 1,19:PRINT STRING$(50," ")
230 LOCATE 1,19:INPUT" Coefficient de co
ntrole (10 a 40) ";k
240 IF (k<10) OR (k>40) THEN SOUND 1,100
,30:GOTO 220
250 CLS
260 INPUT" Barre en degres (RETURN pour
fin) ";ba:b=ba/360*8*ATN(1)
270 IF ba=0 THEN END
280 GOSUB 360
290 GOSUB 450
300 LOCATE 1,1:PRINT STRING$(50," ")
310 LOCATE 1,1:PRINT "Frappez une touche
 pour continuer."
320 r$=INKEY$:IF r$="" THEN 320 ELSE CLS
:GOTO 160
330 '
340 '        CALCUL
350 '
360 n=0:ang=0:va=0
370 mom=k*(b-ang)-f*va
380 aca=mom/in:va=va+aca*dt:ang=ang+va*d
t
390 n=n+1:ang(n)=ang
400 IF n<nm THEN 370
410 RETURN
420 '
430 '        COURBE
440 '
450 ORIGIN 50,20
460 lecr=500:hecr=300:b1=0:b2=5
470 y1=0:y2=1.5
480 ex=lecr/(b2-b1):ey=hecr/(y2-b1)
490 '--------Trace des axes
500 x=b1:PLOT x,1:DRAW x,hecr,1
510 y=y1:PLOT 1,0:DRAW lecr,y
520 '--------Graduation de l'axe OX
530 TAG
540 FOR i=0 TO 5 STEP 2
550   x=100*i:y=-5
560   PLOT x,y:z$=MID$(STR$(i),2):PRINT
z$;" sec.";
```

```
570 NEXT i
580 '--------Graduation de l'axe OY
590 PLOT -50,200:PRINT"    ";
600 PLOT -50,200:z$=MID$(STR$(ba),2):PRI
NT z$;
610 TAGOFF
620 '--------Trace de la courbe
630 FOR x=b1 TO b2 STEP dt
640   y=ang(x/dt)/b
650   sx=(x-b1)*ex:sy=(y-y1)*ey
660   PLOT sx,sy,1
670 NEXT x
680 RETURN
```

Essais de CYBER

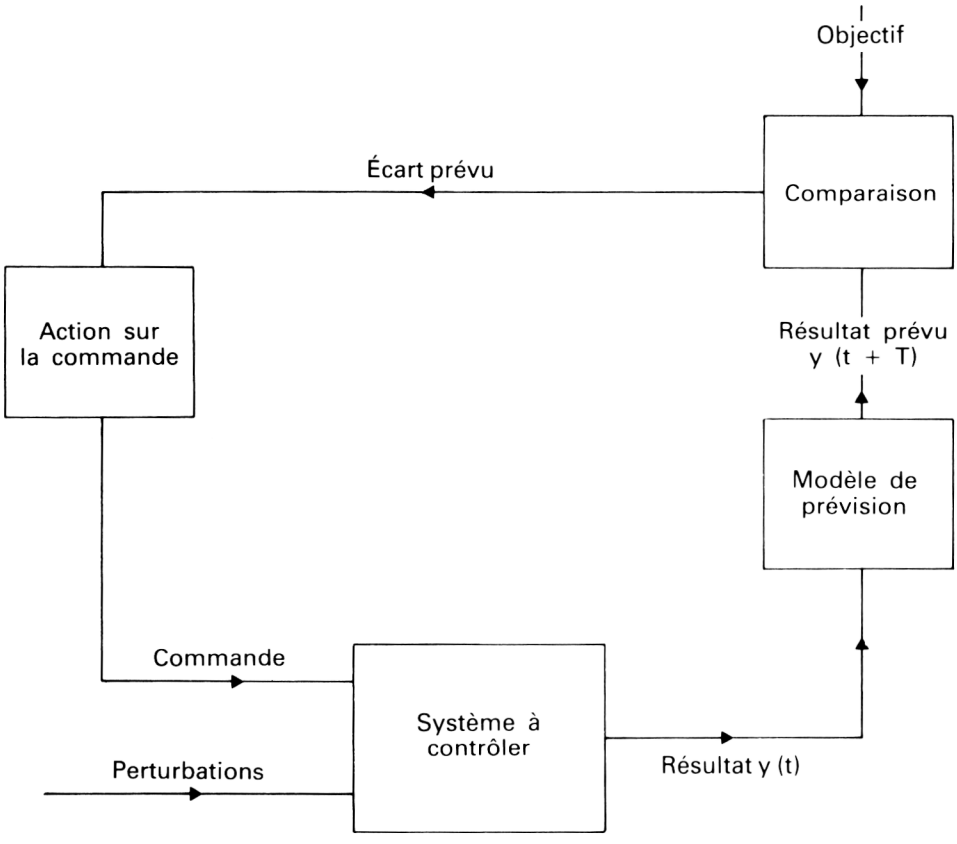
Je vous laisse le plaisir de faire varier les caractéristiques du système et de voir apparaître les courbes associées. Les effets se manifestent sur la fréquence des oscillations, leur amplitude et leur amortissement, jusqu'à l'atteinte de la position d'équilibre.

Vous pourrez encore modifier l'intervalle élémentaire de calcul (variable DT) en lui donnant, par exemple, des valeurs comprises entre 0,01 et 0,1. Pour certaines valeurs de DT, vous le constaterez, le processus "diverge" et les courbes, se ramifiant, perdent toute signification. Il s'agit de problèmes d'analyse numérique hors de notre sujet.

Une autre expérience, assez riche, vaut la peine d'être tentée. Elle donne un aperçu des méthodes utilisées pour la stabilisation des systèmes auto-régulés. L'idée est de remplacer la mesure instantanée du résultat par une **prévision** *(voir le schéma de la page ci-contre).*

Les systèmes réels comportent en effet des inerties (mécaniques, électriques...) qui se traduisent par des délais de réaction. Dans notre expérience simulée, l'inertie principale est celle du gouvernail. Elle tient à sa masse et à sa position par rapport à l'axe de rotation. Une rétroaction liée, comme nous l'avons fait jusqu'ici, à la seule position instantanée du gouvernail, ne tient pas compte de son élan. D'où les oscillations.

Une anticipation est donc souhaitable et sera réalisée par un modèle de prévision. C'est un sous-système dont la fonction est de prévoir la situation y(t+T) du résultat à l'instant t+T, décalé vers l'avenir d'une durée T. Celle-ci est définie, en ordre de grandeur, par le délai global de réaction du système à contrôler. L'écart mesuré, qui agit par feedback, est alors celui qui existe entre l'objectif et la situation extrapolée y(t+T).



Est-ce difficile à réaliser ? Pas le moins du monde. Veuillez, s'il vous plaît, ajouter au programme une ligne 55 :

```
55 T=0.5      (soit un temps de réaction d'une demi-seconde)
```

puis modifier l'instruction de la ligne 330 comme suit :

```
330 MOM=K*(B-ANG*(1+T*VA))-F*VA
```

Refaites alors les simulations. Les résultats sont spectaculaires. Avec DT=0,02, même dans les pires conditions, soit :

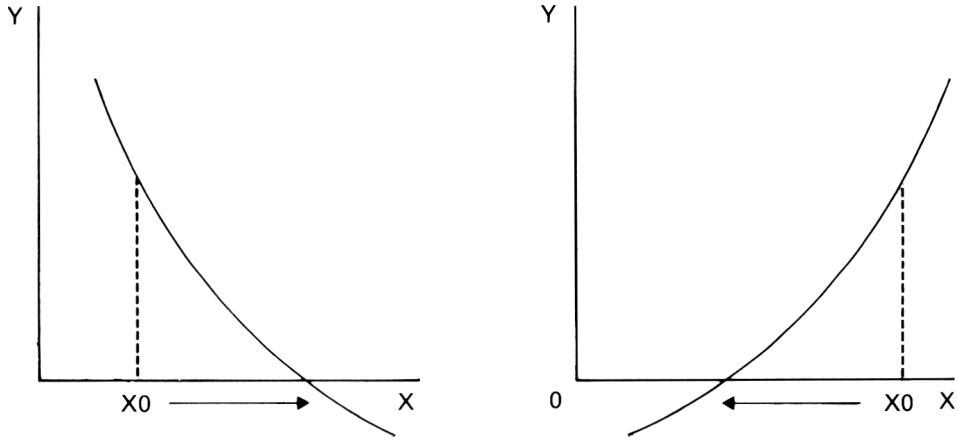
- moment d'inertie : 0,5,
- coefficient de frottement : 1,
- coefficient de contrôle : 20,

le gouvernail atteint sa position finale sans oscillation.
Comparez, je vous prie, à la courbe obtenue dans les mêmes conditions sans utiliser une prévision.

A LA RECHERCHE DU SIGNE D'UNE LIAISON

Tout ceci suppose, nous l'avons répété, une information sur la structure du système : la connaissance du signe ou du sens des liaisons entre certaines commandes et certains résultats. Si nous les ignorons, il est peut-être possible de chercher et d'apprendre.

Prenons, comme simple exemple, la recherche des racines d'une équation (F(X)=0. Partons d'une valeur X0 de la variable. Supposons qu'après calcul, nous trouvions F(X0)>0. Nous devrons, pour cheminer vers la racine, aller dans le sens des X croissants si, autour de X0, la fonction est décroissante, et inversement.



L'information utile est donc la pente de la courbe représentative dans la zone qui avoisine X0. Cette pente P peut être mesurée à partir de deux points voisins. Il suffira alors de donner à X un déplacement de la forme :

$$-K*SGN(P)*(F(X0)-0)$$

où K est un coefficient de contrôle positif à définir.

Tentons, sur ces bases, la construction d'un mini-programme.

Le programme ZERO

Entrez, je vous prie, les instructions suivantes :

```
10 CLS:epsilon=0.000001:k=0.1
20 DEF FNy(x)=(x-1)*(x-3)
30 INPUT"1ere valeur";x0
40 INPUT"2eme valeur";x1
```

```
50 y0=FNy(x0):y1=FNy(x1)
60 IF ABS(y1)<epsilon THEN PRINT:PRINT"R
acine=";x1:END
70 p=(y1-y0)/(x1-x0)
80 x0=x1:x1=x1-k*SGN(p)*y1
90 PRINT USING "#.##########  ";x0,x1
100 GOTO 50
```

La ligne 20 contient la définition de la fonction qui nous intéresse et peut être modifiée *ad libitum.* A partir de deux entrées X0 et X1, le programme calcule une racine, à la précision définie par la variable EPSILON. Pour l'exemple choisi :

F(X)=(X−1)*(X−3)

le programme donne :

Entrées	*Racine*
X0=2:X1=4	Racine=3.00000047

La méthode de Descartes, pour le problème qui nous occupe, utilise la formule :

X1=X1−Y1/P

suggérée par l'assimilation de la courbe à une droite au voisinage du point X1. Mais, vous le voyez, ce n'est qu'une parmi l'infinité des formules possibles. Une condition essentielle : déterminer et respecter le sens (ou le signe) du déplacement.

Le "mécanisme" fonctionne cependant sous réserve de **stabilité** des oscillations autour de la racine. Nous pouvons explorer ce domaine en faisant varier K pour diverses fonctions. Dans notre exemple, la rapidité de convergence et la précision s'améliorent avec des valeurs croissantes de K. La meilleure, pour la fonction choisie, se situe autour de 0,5. Mais, autour de K=1, les problèmes apparaissent, notamment pour la racine 3 (Overflow, Division par zéro). A ce stade, le processus diverge.

Le principe de rétroaction, avec recherche d'objectif et apprentissage de la structure, permet le traitement d'une très grande variété de problèmes. Si le système étudié comprend de nombreuses variables quantitatives, il sera possible, au cours du processus d'exploration, de conduire une "analyse de régression". Cette technique statistique détermine les coefficients de liaison entre certaines commandes et un ou plusieurs résultats. Le sens des actions est alors connu et les boucles de feedback deviennent efficaces. Nous verrons au chapitre 7 avec quelle ingéniosité Samuel a mis en œuvre ce principe dans un programme célèbre jouant aux dames. Nous présentons ci-dessous une application intéressante mais beaucoup plus simple : la recherche des maxima et des minima (valeurs stationnaires) d'une fonction continue définie par son expression.

Le programme OPTIMUM

Donnons-nous, par son expression analytique, une surface dans l'espace à N dimensions (ici trois nous suffiront). Par exemple :

Z=SIN(X+Y)

ou un hémisphère de rayon 1 et de centre (1,2) :

Z=SQR(1−(X−1)^2−(Y−2)^2)

Une condition simple définit un maximum ou un minimum sur la surface : la cote Z ne varie pratiquement pas lors de déplacements au voisinage de ces points. Ailleurs, sur la surface, le programme mesurera les pentes dans les directions parallèles aux axes, au voisinage de la position du point courant. Selon le précepte de Gide, il suivra ensuite sa pente, montante ou descendante selon les conditions initiales. Mettons cela en œuvre sous la forme du court programme ci-dessous listé :

```
10 CLS:epsilon=0.00000001:k=0.1
20 RANDOMIZE TIME
30 PRINT TAB(11)"RECHERCHE D'OPTIMUM":PR
INT:PRINT
40 DEF FNz(x,y)=SIN(x+y)
50 a$="z(x,y)=SIN(x+y)"
60 PRINT a$:PRINT:PRINT
70 INPUT"1eres valeurs x,y";x0,y0
80 INPUT"2emes valeurs x,y";x1,y1
90 z0=FNz(x0,y0):z1=FNz(x1,y1)
100 IF ABS(z1-z0)<epsilon THEN PRINT:PRI
NT "Optimum approche: Z=";ROUND(z1,3):PR
INT:PRINT "X=";ROUND(x1,1),"Y=";ROUND(y1
,1):END
110 px=(z1-z0)/(x1-x0):py=(z1-z0)/(y1-y0
)
120 x0=x1:x1=x1+(k+RND)*SGN(px)*(z1-z0)
130 y0=y1:y1=y1+(k+RND)*SGN(py)*(z1-z0)
140 GOTO 90
```

Au départ, entrez deux couples de coordonnées X,Y. A partir d'elles, l'instruction 100 calcule les pentes. Les instructions 120 et 130 déterminent le déplacement du point courant selon les deux axes du plan XOY, à partir du signe des pentes et de l'écart de cote. L'introduction de nombres aléatoires évite de limiter les mouvements aux seules directions parallèles aux diagonales du plan. Le coefficient K de contrôle peut être ajusté pour expérimentation. De même, vos propres définitions de fonctions peuvent évidemment remplacer celle de l'instruction 30.

Le choix d'une surface aussi bosselée que celle associée à la fonction Z=SIN(X+Y) est, *a priori*, susceptible d'affoler le plus avisé des ordinateurs. Le vôtre s'en tire apparemment bien et l'écran affiche, par exemple, ceci :

```
                    RECHERCHE D'OPTIMUM

z(x,y)=SIN(x+y)

1eres valeurs x,y? 0.2,0.36
2emes valeurs x,y? 0.75,0.4

Optimum approche: Z= 0.998

X= 1            Y= 0.6
```

La précision de calcul est, pour être honnête, souvent moins bonne que celle de l'exemple. Les optima, chacun le sait, sont généralement plats et leur sommet ne peut être déterminé, par cette méthode, avec une grande exactitude.

UNIVERSALITÉ DES RÉTROACTIONS

Quelques rares exemples de mécanismes auto-régulés se rencontrent dans le lointain passé (régulateurs de Watt), mais leur développement exponentiel date de la Seconde Guerre mondiale et, notamment, des travaux de Norbert Wiener. A cette date, la théorie était établie et les applications, aidées par la naissante électronique, se multiplièrent.

Les thermostats, amplificateurs radio, servo-mécanismes de direction assistée, pilotes automatiques d'avions en roulis et en tangage sont des exemples bien connus de machines faites de main d'homme qui, grâce à la rétroaction, atteignent leurs objectifs de manière autonome, en dépit des perturbations. Un radar de poursuite est maintenu pointé sur l'avion à suivre malgré l'immense variété des trajectoires possibles et des perturbations touchant l'amplitude de l'écho radar, le rayonnement du ciel, les bruits thermoioniques dus à l'incessante danse des électrons, le vent sur l'antenne, etc. Ni les unes

ni les autres ne sont d'ailleurs bien connues et, cependant, l'ensemble fonctionne avec une précision d'apparence miraculeuse.

Moins connus sont les mécanismes de régulation biologique. Pourtant, toutes les synthèses, tous les équilibres chimiques, posturaux, les tropismes, les activités sensorielles et motrices sont réglés par des circuits à feedback.

Donnons deux exemples, choisis parmi des milliers.

Le taux de calcium dans le sang (calcémie) est maintenu par des mécanismes contrôlant les vitesses de destruction et de reformation des os. Lorsqu'il descend en dessous d'un certain seuil, l'écart correspondant déclenche la sécrétion, par la parathyroïde, d'une hormone qui effectue la lyse (décalcification) des os et met en circulation, dans le sang, le calcium ainsi libéré. Le processus inverse (synthèse des os) est sous le contrôle de la thyroïde.

La contraction d'un muscle, en réponse à une stimulation d'un organe sensoriel, est un processus à deux phases. Lors de la première, se contractent des fibres spéciales qui n'exercent aucune tension. La deuxième phase aligne par feedback (lié à l'écart de longueur), sur ces fibres spéciales jouant le rôle d'objectifs intermédiaires, les fibres principales qui exercent les efforts effectifs.

Vous me permettrez peut-être, ami(e) lecteur(trice), de clore ce chapitre sur deux remarques philosophiques.

Il était (et est encore parfois) admis que les mondes inanimé et vivant se distinguent sur un point essentiel : la finalité. Or il n'en est rien et de façon patente. Des mécanismes physiques, naturels ou artificiels, à travers les rétroactions, poursuivent, atteignent et maintiennent les objectifs ancrés dans leur structure profonde.

Mieux encore, un système très complexe, composé de multiples éléments en interaction, crée naturellement des objectifs, au moins en apparence. La raison en est simple : parmi un ensemble de liaisons entre variables nombreuses, apparaissent, en toute probabilité, des boucles ou circuits. Selon le sens des actions, les variables intéressées divergent, explosent, avec changement de structure de leur environnement ou, au contraire, se maintiennent, contre vents et marées, autour de valeurs d'équilibre. Le chapitre suivant vous en dira davantage.

Chapitre 3
L'ÉTRANGE MACHINE D'ASHBY ET LES AUTRES

Le chapitre précédent nous a montré l'étendue et l'efficacité de la rétroaction. La méthode a cependant ses limites. Les problèmes évoqués ont pu être résolus car ils présentent une particularité de structure : commandes et résultats sont des variables quantitatives ou, tout au moins, ordonnées. Cette caractéristique permet, après divers essais, de déterminer le sens de certaines interactions et d'en tirer profit. Mais il n'en est, évidemment, pas toujours ainsi.

Prenons l'exemple d'un pilote automatique d'avion en roulis. Il fonctionne dans son environnement normal. Supposons que, par suite d'un défaut de fabrication ou pour toute autre raison, certaines connexions viennent à s'inverser dans les circuits électriques de commande. Les actions de l'organe de contrôle s'exercent alors à contre-sens, amplifiant les perturbations jusqu'à destruction de l'avion, sauf modification du processus.

L'intervention d'un pilote humain est la forme classique d'une telle modification mais nous pouvons nous demander s'il existe des "mécanismes" plus généraux, capable de remplacer l'intervention intelligente ou de la simuler.

L'ADAPTATION DES ORGANISMES

Un très jeune chaton évolue dans une salle à manger, près d'une cheminée où flambe un feu vif. Il se dirige hardiment vers le tablier. Au milieu des crépitements, un charbon ardent roule hors du foyer. Le chaton y pose sa patte, crie de douleur et court en tout sens.

Quelques jours après, il a déterminé sa place, à distance suffisante du foyer. Il sait s'en rapprocher si le feu décline et s'en éloigner lorsque dansent les flammes.

Les phénomènes quotidiens de ce type manifestent la capacité d'adaptation des êtres vivants, condition fondamentale de leur survie. Une question vient immédiatement à l'esprit : l'adaptation est-elle une propriété spécifique de la vie ? Pouvons-nous, au contraire, construire des machines adaptatives et, si oui, quels en seront les rouages ?

A l'aube de la cybernétique, de nombreux chercheurs ont tenté d'explorer ce domaine. En 1952, parut la première édition d'un ouvrage du biologiste britannique Ross Ashby. Le titre *Design for a brain (Projet pour un cerveau)* en décrit assez l'ambition.

Première étape : définir, de manière précise, le problème à résoudre. Prenons comme exemple, pour un animal, l'état de "vivant" par opposition à "mort". Pour qu'un animal reste vivant, suggère Ashby, certaines variables, qualifiées d'essentielles, doivent rester à l'intérieur de limites physiologiques. La nature et les bornes de ces variables sont fixées à l'intérieur de chaque espèce. La température du corps, la pression sanguine, le rythme de la respiration, le nombre de globules blancs et rouges, les concentrations de certaines substances chimiques, les intensités de plaisir et de douleur, etc., sont de telles variables essentielles. Leur nombre, pour un mammifère, avoisine la centaine ou davantage.

D'un point de vue formel, vivre c'est maintenir les variables essentielles à l'intérieur de leurs limites et ce, dans des contextes et environnements variables, calmes ou dangereux. Si une situation donnée entraîne l'une au moins des variables essentielles hors limites (brûlure du charbon ardent pour notre chaton), l'organisme adaptatif réagit, modifie ses comportements, l'environnement ou même sa structure interne jusqu'au retour à des conditions normales.

Notez, je vous prie, la similitude de ces concepts avec la définition générale d'un problème (*cf.* chapitre 1). Résoudre un problème implique la construction de solutions possibles et la vérification des conditions posées. S'adapter, c'est générer des comportements variés jusqu'au retour de toutes les variables essentielles à l'intérieur de leurs bornes.

Quelle pourrait être, s'est demandé Ashby, la structure générale **la plus simple** capable d'adaptation ? Le système imaginé comporte un premier niveau de régulation, formé de boucles de rétroaction liant diverses variables. Il assure le maintien de certains objectifs internes. Ceci correspond, pour fixer les idées au niveau des réflexes.

Si les perturbations extérieures sont fortes ou d'une autre nature que celles neutralisables par les boucles intérieures de rétroaction, le système peut modifier des paramètres de structure. Il le fait par l'intermédiaire d'un **générateur de variété**. Celui-ci est déclenché si l'une au moins des variables essentielles du système sort de ses limites

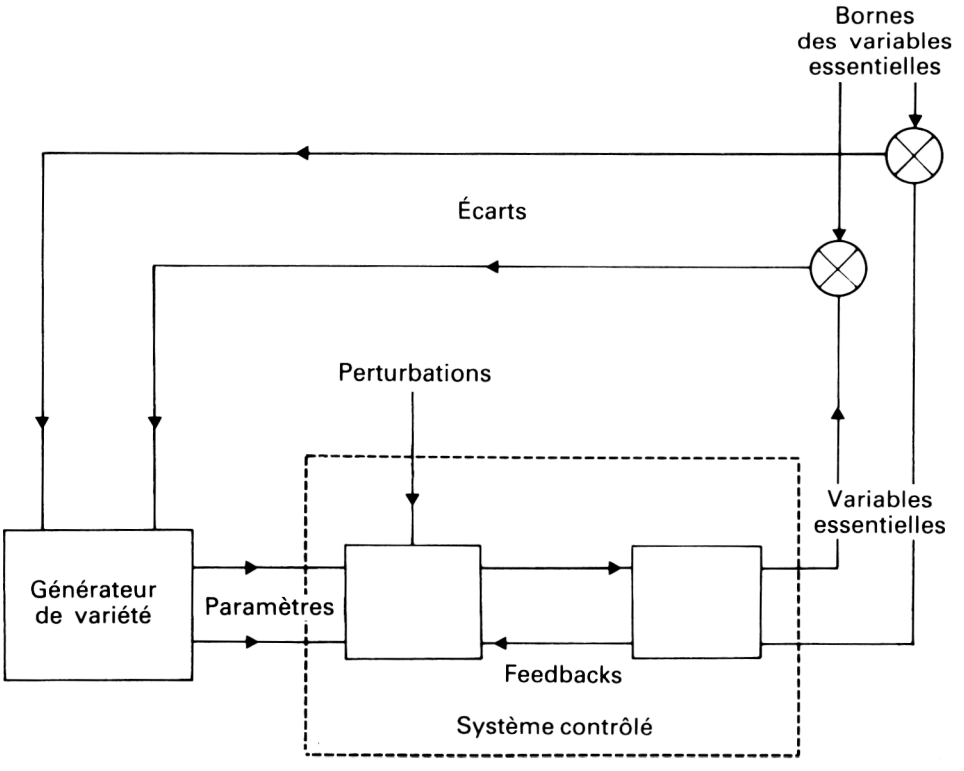
naturelles. Il modifie alors, selon une loi définie mais arbitraire, éventuellement aléatoire, un ou plusieurs paramètres. En termes plus familiers, il cherche à tâtons.

Dans le contexte ainsi transformé, le système évolue et deux cas peuvent alors se présenter :

- si toutes les variables essentielles reviennent à l'intérieur de leurs plages d'objectifs, le générateur de variété est bloqué et le système continue à fonctionner dans la configuration qui vient d'être créée, avec l'aide de ses boucles de rétroaction ;
- si, au contraire, l'une au moins des variables essentielles reste à l'extérieur des limites, le générateur de variété modifie, une nouvelle fois, les paramètres et permet au système de fonctionner pendant une certaine durée dans cette autre configuration.

Cette procédure est répétée jusqu'à atteinte d'un équilibre d'ensemble si, toutefois, un tel équilibre peut être obtenu de cette manière.

Ashby qualifie d'"ultra-stables" les systèmes possédant cette propriété. Leur structure fonctionnelle est symbolisée par le graphique suivant :



Graphique fonctionnel d'un système ultra-stable

L'HOMÉOSTAT D'ASHBY ET SON ÉQUIVALENT SUR AMSTRAD

Il convenait d'incarner la théorie précédente dans une machine concrète, disponible pour l'expérimentation. Ashby n'y manqua pas. Il construisit un modèle électrique (en ce temps-là, les ordinateurs faisaient cruellement défaut) baptisé "HOMEOSTAT" et dont voici une description succincte.

L'HOMÉOSTAT se compose de quatre unités, au sommet desquelles pivotent quatre aimants (un par unité). Les déviations angulaires des aimants par rapport à leurs positions centrales représentent les quatre variables essentielles du système.

Chaque unité émet un courant continu proportionnel à l'écart angulaire de l'aimant à partir de sa position centrale. Les unités sont reliées ensemble de sorte que chacune d'elles envoie aux trois autres son courant de sortie et reçoive, en entrée, les courants émis par les trois autres. Dans chaque unité, les courants reçus s'ajoutent dans une bobine magnétique qui commande, à son tour, la position de l'aimant.

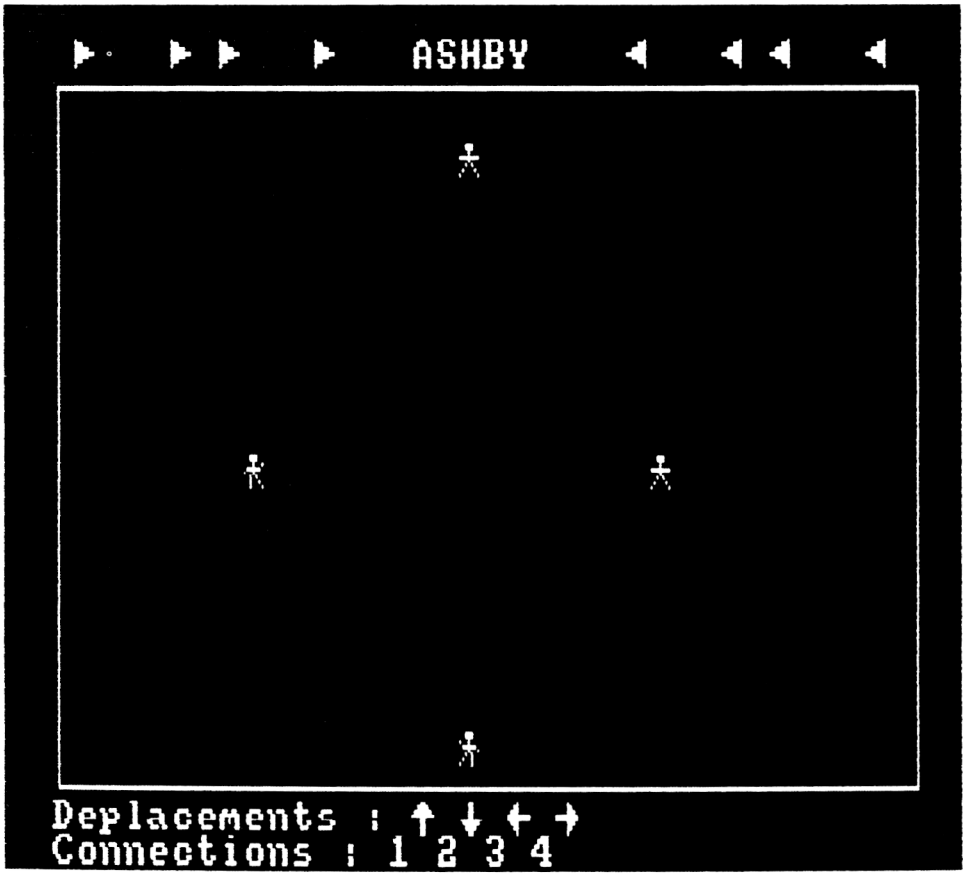
Certains composants (inverseurs et potentiomètres) des circuits électromagnétiques de chaque unité peuvent être commutés par un sélecteur, lui-même lié à l'intensité du courant de sortie. Ces composants constituent les paramètres. Si l'un des aimants vient en butée aux extrémités de sa course (variable essentielle hors limites), le sélecteur est activé. Il commute alors effectivement les composants, modifiant ainsi les liaisons électriques entre les quatre unités. L'équilibre du système est atteint lorsque les quatre aimants sont en position centrale.

L'expérience consiste à ajuster les positions et la structure interne initiales du système puis à observer son comportement en recherche d'équilibre.

Le programme ASHBY

Nous avons l'avantage de disposer d'une merveilleuse machine, capable de simuler toutes les autres : notre micro-ordinateur. Aussi n'avons-nous pu résister longtemps à l'envie d'écrire un programme qui simule, en quelques lignes de BASIC, l'HOMÉOSTAT d'Ashby.

Remplaçons les quatre aimants par quatre petits bonshommes qui se déplacent sur l'écran, chacun le long des côtés d'un carré, comme l'indique la copie d'écran.



Leurs positions sont repérées par la distance, positive ou négative, à la position centrale, distance comptée en nombre de cases (une case = 1 caractère). Elles sont mémorisées dans la table d'écarts E(4). Définissons alors une "matrice de couplage" F(I,J) formée de coefficients placés au croisement des lignes de rang I et des colonnes de rang J, dans une table de dimensions (4,4). A intervalles périodiques, seront calculés, pour chaque position d'indice I, des déplacements D(I) :

D(I)=D(I)+F(I,J)*E(J)

Ils modifieront les écarts E(I) par :

E(I)=E(I)+D(I)

Le sous-programme "EVOLUTION DU SYSTEME" exprime cela (lignes 370 à 420).

```
260 '
270 '          EVOLUTION DU SYSTEME
280 '
290 IF INKEY(0)=0 THEN d(1)=d(1)+5
```

```
300 IF INKEY(8)=0 THEN d(2)=d(2)+5
310 IF INKEY(2)=0 THEN d(3)=d(3)+5
320 IF INKEY(1)=0 THEN d(4)=d(4)+5
330 IF INKEY(13)=0 THEN f(1,1)=-f(1,1)
340 IF INKEY(14)=0 THEN f(2,1)=-f(2,1)
350 IF INKEY(5)=0 THEN f(3,1)=-f(3,1)
360 IF INKEY(20)=0 THEN f(4,1)=-f(4,1)
370 FOR i=1 TO n
380   FOR j=1 TO n
390     d(i)=d(i)+f(i,j)*e(j)
400   NEXT j
410   f=0:GOSUB 530:'Affichage
420   e(i)=e(i)+d(i)
430   IF ABS(e(i))>=ma AND (TIME-t!)>600
 THEN t!=TIME:GOSUB 610:' Reorganisation
440   IF e(i)>=ma THEN e(i)=ma
450   IF e(i)<=-ma THEN e(i)=-ma
460   f=1:GOSUB 530:' Affichage
470   d(i)=0
480 NEXT i
490 GOTO 290
```

Si le système est instable, l'une au moins des figurines viendra buter contre les bords du cadre. L'écart associé dépassera la valeur maximale MA, fixée ici à 10 unités, déclenchant ainsi l'appel au sous-programme "REORGANISATION" :

```
580 '
590 '          REORGANISATION
600 '
610 ENT 2,=68,20,=95,20,=68,20,=0,0
620 SOUND 1,0,100,15,,2
630 FOR j=2 TO n
640   IF RND<0.5 THEN f(i,j)=-RND ELSE f
(i,j)=RND
650 NEXT j
660 RETURN
```

qui tire au sort, à nouveau, les coefficients des colonnes 2 à 4 de la matrice de couplage avec un signe aléatoire.

L'instant de cette opération est mémorisé dans une variable T. Le système dispose de deux secondes pour vérifier la stabilité de la nouvelle structure. Un sous-programme d'affichage est appelé deux fois dans le cycle de calcul, l'une pour écrire (drapeau F=1), l'autre pour effacer (drapeau F=0).

Les instructions 290 à 360 permettent l'exécution de commandes au clavier. Un appui sur l'une des touches fléchées déclenche un déplacement de 5 unités de la figurine associée et permet ainsi un test de stabilité de chaque configuration.

Le programme transforme en inverseurs les touches 1 à 4 du clavier numérique. Un appui sur l'une de ces touches change le signe des coefficients F(1,1) à F(4,1) qui restent à l'abri des restructurations aléatoires. Les instructions non encore mentionnées tracent un cadre sur l'écran, construisent la matrice initiale avec des coefficients tirés au hasard et affichent les positions des figurines.

Maintenant, entrez, s'il vous plaît, le programme ASHBY (la tâche n'est pas démesurée) et lancez-le. Le temps des expériences est venu.

Au départ, nos quatre bonshommes restent gentiment à leurs positions d'équilibre, dessinant un losange sur l'écran. Cet équilibre peut être stable ou instable. Pour le savoir, nous appuyons (franchement) sur l'une des touches fléchées. Deux cas possibles :

– *équilibre stable :* les personnages exécutent quelques pas de danse, sous l'action des forces de rappel. Le retour à l'équilibre peut être vif, avec des oscillations progressivement amorties, ou, au contraire, très lent. L'appui sur les autres touches fléchées permet de vérifier la stabilité générale.

– *équilibre instable :* l'une au moins des figurines vient en butée. Dès lors, se déclenchent des mouvements variés, entrecoupés d'immobilités pouvant provoquer l'attente anxieuse de l'opérateur. Puis, brusquement ou lentement, le retour à l'équilibre survient.

Partons d'une telle position d'équilibre stable et livrons-nous à une expérience de neurophysiologie sur ordinateur. Inspirons-nous des travaux de Marina et Sperry qui, sur des singes, coupaient les attaches de certains muscles d'un œil ou d'un membre pour les rattacher ensuite en position inversée. A leur surprise, après quelque temps, la coordination des mouvements se rétablissait, rendant à nouveau possible la vision binoculaire et la marche normale.

Notre homéostat peut le faire ! Pressons l'une des touches 1 à 4, inversant ainsi une connection, puis appuyons sur une des touches fléchées pour vérifier l'équilibre. Si celui-ci est maintenu, insistons avec de nouvelles touches jusqu'à l'instabilité. Le système, ainsi perturbé, va rechercher et retrouver une position stable.

Procédons maintenant à une expérience de "dressage". Les positions des figurines sont numérotées ainsi :

1
2 4
3

Appuyons sur la touche fléchée associée à 4 (celle de droite) et supposons que le déplacement initial du 3e bonhomme s'effectue à droite. Nous voulons que notre partenaire n° 3 réagisse à gauche.

Sanctionnons-le en pressant l'une ou l'autre des touches fléchées assez longtemps pour provoquer un déséquilibre. Après retour à la stabilité, recommençons, si nécessaire, jusqu'au succès. L'homéostat a trouvé la structure convenable et satisfera notre désir jusqu'à nouvel ordre.

Vous pouvez aussi, si vous êtes curieux, modifier le délai d'attente entre restructurations et porter sa valeur à 100 (ou, au contraire, à 1200 et plus), modifier l'amplitude maximale MA des écarts, changer les connexions inversables ou, mieux, imaginer d'autres systèmes ultra-stables. Le champ n'est pas limité. Ah ! j'oubliais. Quand vous en aurez assez, appuyez deux fois sur la touche "Escape" pour arrêter le programme.

Pour clore le sujet, voici le listing complet du programme ASHBY, précédé de la liste des variables.

Liste des variables

Tables :

C(4)	: colonnes d'affichage.
D(4)	: déplacements des figurines.
E(4)	: positions des figurines.
F(4,4)	: matrice de couplage.
L(4)	: lignes d'affichage.

Variables simples :

F	: drapeau pour affichage.
H	: pour alterner les codes des figurines.
I,J	: indices de boucles.
MA	: écart maximal.
N	: nombre de figurines.
P1, P2, P3, P4	: dimensions du cadre d'écran.
T	: horloge.

Listing du programme ASHBY

```
10 ON BREAK GOSUB 670
20 MODE 1:PAPER 10:PEN 0:CLS
30 FOR i=2 TO 16
40 LOCATE i,2:PEN (i MOD 10):PRINT CHR$(
246)
50 LOCATE 41-i,2:PEN(i MOD 10):PRINT CHR
$(247)
60 NEXT
70 PEN 0
```

```
80 LOCATE 18,2:PRINT "ASHBY"
90 PLOT 36,36,1
100 DRAW 610,36,0:DRAW 610,360,0:DRAW 36
,360,0:DRAW 36,36,0
110 LOCATE 3,24:PRINT "Deplacements : ";
120 FOR i=240 TO 243:PRINT CHR$(i);" ";:
NEXT
130 LOCATE 3,25:PRINT "Connections : 1 2
 3 4"
140 ' ---------Initialisation
150 DEFINT a-c,g-z
160 MA=8:p1=5:p2=11:p3=22:p4=28
170 h=0:l(1)=p1:c(2)=p2:l(3)=p3:c(4)=p4:
n=4
180 RANDOMIZE TIME
190 FOR i=1 TO n
200   FOR j=1 TO n
210     IF RND<0.5 THEN f(i,j)=-RND ELSE
 f(i,j)=RND
220   NEXT j
230 NEXT i
240 GOSUB 290:' Evolution du systeme
250 END
260 '
270 '        EVOLUTION DU SYSTEME
280 '
290 IF INKEY(0)=0 THEN d(1)=d(1)+5
300 IF INKEY(8)=0 THEN d(2)=d(2)+5
310 IF INKEY(2)=0 THEN d(3)=d(3)+5
320 IF INKEY(1)=0 THEN d(4)=d(4)+5
330 IF INKEY(13)=0 THEN f(1,1)=-f(1,1)
340 IF INKEY(14)=0 THEN f(2,1)=-f(2,1)
350 IF INKEY(5)=0 THEN f(3,1)=-f(3,1)
360 IF INKEY(20)=0 THEN f(4,1)=-f(4,1)
370 FOR i=1 TO n
380   FOR j=1 TO n
390     d(i)=d(i)+f(i,j)*e(j)
400   NEXT j
410   f=0:GOSUB 530:'Affichage
420   e(i)=e(i)+d(i)
430   IF ABS(e(i))>=ma AND (TIME-t!)>600
 THEN t!=TIME:GOSUB 610:' Reorganisation
440   IF e(i)>=ma THEN e(i)=ma
450   IF e(i)<=-ma THEN e(i)=-ma
460   f=1:GOSUB 530:' Affichage
470   d(i)=0
480 NEXT i
490 GOTO 290
500 '
```

```
510 '          AFFICHAGE
520 '
530 IF i=1 OR i=3 THEN c(i)=(p2+p4)/2+IN
T(e(i))
540 IF i=2 OR i=4 THEN l(i)=(p1+p3)/2+IN
T(e(i))
550 LOCATE c(i),l(i)
560 IF f=1 THEN PRINT CHR$(249+h):h=(h+1
) MOD 3 ELSE PRINT " "
570 RETURN
580 '
590 '          REORGANISATION
600 '
610 ENT 2,=68,20,=95,20,=68,20,=0,0
620 SOUND 1,0,100,15,,2
630 FOR j=2 TO n
640   IF RND<0.5 THEN f(i,j)=-RND ELSE f
(i,j)=RND
650 NEXT j
660 RETURN
670 PAPER 0:PEN 1:CLS:END
```

LES SYSTÈMES COMPLEXES

L'homéostat d'Ashby, simulé sur Amstrad, a mis en évidence ceci : *des éléments simples, unis par un petit nombre de relations, peuvent engendrer des systèmes complexes dont le comportement défie souvent l'analyse.*

En voici un autre exemple qui pourra, de plus, enrichir votre écran de beaux graphismes.

Les automates cellulaires

Ils sont actuellement très étudiés. Un automate cellulaire est composé d'éléments, tous identiques, appelés cellules. Chacune d'elles évolue sous l'influence de règles simples, mais l'assemblage de cellules en "colonies" et l'évolution de ces colonies donnent naissance à des structures d'une extraordinaire complexité.

Nous limiterons notre exemple aux automates cellulaires à une dimension. A chaque stade de l'évolution d'une colonie, toute cellule est dans l'un des états d'une liste définie. Des règles simples permettent, à chaque génération, le calcul du nouvel état de chaque cellule en fonction de son état précédent et des états des cellules voisines.

Supposons trois états : 0, 1, 2. L'état suivant d'une cellule sera déterminé en faisant la somme des valeurs liées à son état actuel et à celui de ses deux voisines immédiates, à gauche et à droite. Définissons ensuite une correspondance entre cette somme et l'état suivant, par exemple :

Somme	*Etat suivant*
0	0
1	1
2	2
3	2
4	0
5	1
6	2

Ceci fait, la donnée des états initiaux des cellules de notre colonie définit complètement toute l'évolution ultérieure.

Traduisons immédiatement en un programme sans prétention qui mettra en évidence la puissance des instructions graphiques de l'Amstrad.

Comme l'indique la copie d'écran ci-dessous, nous définissons l'état initial de la colonie (40 cellules sur une ligne d'écran) par :

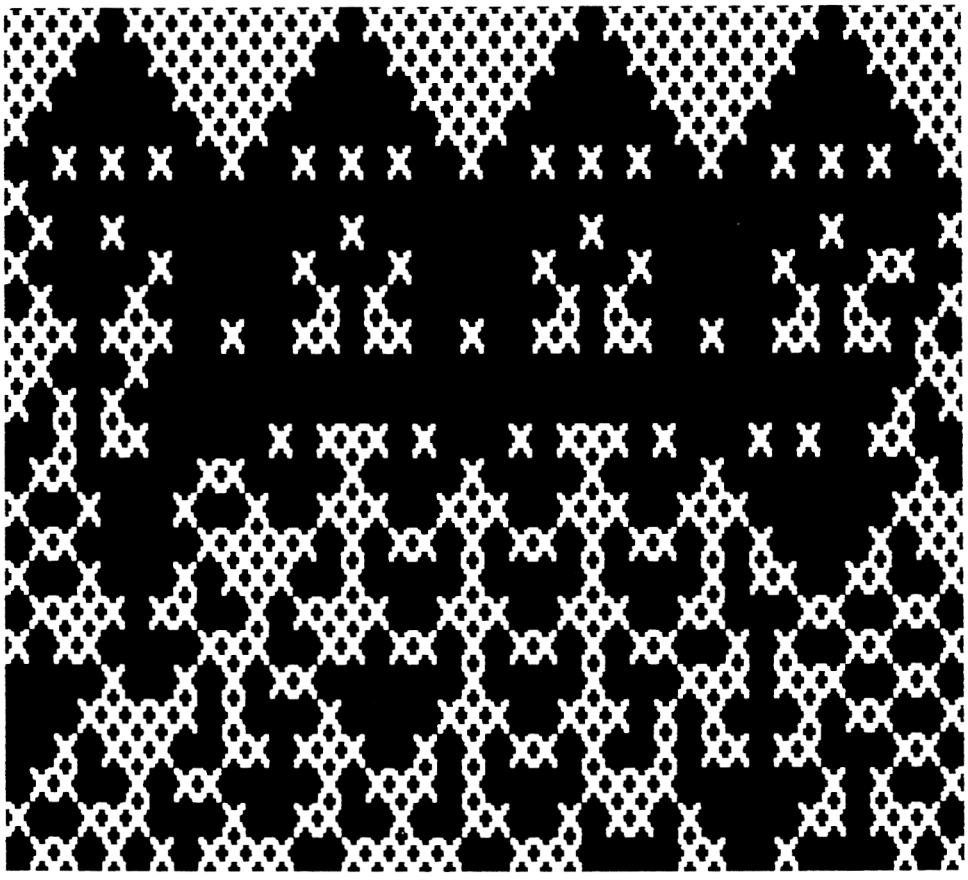
– le nombre de germes d'une "couleur" différente de 0 ;
– leurs positions (entre 1 et 40) et couleurs (1 ou 2).

```
                    CELLULES

 Nombre de germes? 4

 Tirage aleatoire (o/n) n

Entrez les positions des germes(1 a 40)
puis leur couleur(1 ou 2)
en separant les deux par une virgule.
En fin de programme frapper une touche.

Germe no. 1 Position,couleur? 5,1
Germe no. 2 Position,couleur? 15,2
Germe no. 3 Position,couleur? 25,2
Germe no. 4 Position,couleur? 35,1
```

A partir des multiples configurations possibles, le programme "CELLULES" génère une colonie qui emplit l'écran. La copie imprimée ci-dessous (négatif en noir et blanc), réalisée avec 4 germes en positions

5, 15, 25, 35 et couleur 1, ne donne qu'une faible idée des graphismes réalisables, en particulier sur écran couleur. Essayez par exemple :

– 1 germe en position 20, couleur 2 ;
– 2 germes en positions 10 et 30, couleur 2.



Le programme CELLULES

En voici le listing, précédé d'une liste de variables.

Liste des variables

Tables :

PG(NG) : positions des germes (NG=nombre de germes).
CG(NG) : couleurs des germes.

Variables simples :

C0 : numéro de colonne.
I,J : indices de boucle.
LI : numéro de ligne.
NG : nombre de germes.
S : somme des couleurs d'une cellule et de ses deux voisines.
X,Y : coordonnées du centre d'une cellule.
R$: réponse à une question.
Z$: caractère affiché dans chaque cellule.

Listing du programme CELLULES

```
10 ' -------- Initialisation
20 MODE 1:PAPER 3:PEN 1:z$=CHR$(203)
30 DEFINT a-z:RANDOMIZE TIME:CLS
40 LOCATE 16,2:PRINT "CELLULES":PRINT
50 ' -------- Definition des germes
60 INPUT" Nombre de germes";ng:DIM pg(ng
),cg(ng)
70 PRINT:PRINT " Tirage aleatoire (o/n)
";
80 WHILE r$<>"o" AND  r$<>"n"
90 r$=LOWER$(INKEY$)
100 WEND
110 PRINT r$
120 IF r$="o" THEN 490
130 PRINT:PRINT "Entrez les positions de
s germes(1 a 40)"
140 PRINT "puis leur couleur(1 ou 2)"
150 PRINT "en separant les deux par une
virgule."
160 PRINT "En fin de programme frapper u
ne touche.":PRINT
170 FOR i=1 TO ng
180   PRINT "Germe no.";i;:INPUT"Positio
n,couleur";pg(i),cg(i)
190   IF pg(i)<1 OR pg(i)>40 OR cg(i)<1
OR cg(i)>2 THEN 180
200 NEXT i
210 ' -------- Evolution des cellules
220 CLS
230 FOR i=1 TO 40
240   PEN 0
250   FOR j=1 TO ng
260     IF pg(j)=i THEN PEN cg(j)
270   NEXT j
```

```
280    PRINT z$;
290 NEXT i
300 FOR li=2 TO 25
310    FOR co=1 TO 40
320      x=(co-1)*16+8:y=400-(li-1)*16+8
330      s=TEST(MAX(8,x-16),y)+TEST(x,y)+
TEST(MIN(632,x+16),y)
340     IF s=0 OR s=4 THEN PEN 0
350     IF s=2 OR s=3 OR s=6 THEN PEN 2
360     IF s=1 OR s=5 THEN PEN 1
370     LOCATE co,li:PRINT z$;
380   NEXT co
390 NEXT li
400 ENT 2,=45,15,=132,14,=98,15,=156,20,
=127,15
410 FOR i=1 TO 3:SOUND 1,0,30,15,,2:NEXT
420 ' -------- Sortie du programme ou rep
rise
430 IF INKEY$="" THEN 430
440 PAPER 0:PEN 1:CLS
450 PRINT "Nous continuons ? ";
460 r$=LOWER$(INKEY$)
470 IF r$="" OR (r$<>"o" AND r$<>"n") TH
EN 460
480 IF r$="o" THEN PRINT r$:CLEAR:GOTO 2
0 ELSE PRINT r$:END
490 '
500 ' Tirage aleatoire
510 '
520 FOR i=1 TO ng
530    pg(i)=INT(40*RND+1)
540    cg(i)=INT(2*RND+1)
550 NEXT
560 GOTO 220
```

Commentaires

L'opérateur définit les positions et couleurs des germes (lignes 50 à 200). Les instructions 230 à 290 construisent ensuite la première ligne d'écran : elles placent les germes et impriment leurs couleurs. Les autres caractères des cellules de la première ligne sont affichés en noir (PEN 0).

Les lignes 300 à 390 calculent le contenu de chaque cellule courante. Elles utilisent la fonction TEST pour repérer les couleurs de la cellule placée immédiatement au-dessus et de ses deux voisines. Les fonctions MAX et MIN sont liées à l'incidence des bords de l'écran.

Applications des automates cellulaires

Les êtres de ce type sont observables sous deux angles :

- systèmes réels ou modèles de tels systèmes ;
- systèmes informatiques.

Les sciences physiques traditionnelles reconnaissaient à leurs frontières des problèmes qui défient l'analyse mathématique : les écoulements de fluides en régime turbulent, par exemple. Les équations de l'hydrodynamique sont parfaitement connues. Cependant, le calcul direct, à partir d'elles, des trajectoires des gouttes, filets et torrents qui coulent dans votre lavabo est le treizième travail d'Hercule. Seules, l'observation directe et la simulation pas à pas sont utilisables.

Or les automates cellulaires, à l'exception de quelques systèmes très simples, présentent cette propriété. Soit la question : "Est-il possible, sans utiliser de simulation pas à pas, de **calculer à l'avance** l'état d'une colonie, après N étapes, à partir des données initiales et des règles d'évolution ?"

La réponse est généralement négative. En d'autres termes, il n'existe pas de court-circuit que pourrait mettre en œuvre un système plus puissant que l'automate cellulaire considéré. Celui-ci est dit **irréductible.**

Autre question liée à la première : "Les cellules de l'automate seront-elles toutes de la même couleur (noire dans notre exemple) au bout d'un temps fini ?"

Pas de réponse ! Je vais insister car vous pourriez comprendre que la réponse n'est pas encore connue mais qu'elle pourra l'être un jour. Dans le cas général, il n'y aura jamais de réponse. La question est **"indécidable".** Le calcul est impossible et l'on ne pourra savoir, en cas d'échec, si une éventuelle simulation a été poussée assez loin.

En 1931, un très grand logicien, Kurt Gödel démontra un théorème prouvant l'existence, dans tout système mathématique contenant l'arithmétique, d'énoncés qui ne sont ni démontrables ni réfutables en un nombre fini d'étapes. Ces énoncés pouvaient apparaître comme des monstres complexes et très exceptionnels. Le programme "CELLULES" en met un à portée de votre clavier.

Les sciences de la nature ont surtout analysé des phénomènes réductibles, donc calculables à partir des lois, mais plusieurs raisons donnent à penser que ceux-ci sont l'exception plutôt que la règle. Il se peut, par exemple, que l'évolution et la forme finale des organismes biologiques à partir de leur code génétique ne puissent être déterminées qu'en suivant, pas à pas, leur développement réel. D'où l'intérêt porté à l'étude des automates cellulaires qui sont les plus simples des systèmes irréductibles et peuvent servir de modèles à quelques autres.

Sous un angle différent, un automate cellulaire est un système informatique analogue à un ordinateur. La suite des valeurs initiales

des cellules peut être vue comme un ensemble de données codées. Cette information est, au cours de l'évolution, "traitée" selon les règles définies.

Les ordinateurs numériques sont des machines universelles capables de n'importe quel algorithme. Qu'en est-il de nos automates cellulaires ?

Plusieurs d'entre eux, on le sait, sont des calculateurs universels. L'un deux est l'automate cellulaire à deux dimensions dont les cellules peuvent prendre deux états, 0 et 1. Il semble, par ailleurs, qu'un assez grand nombre d'automates cellulaires à une dimension soient également des calculateurs universels donc susceptibles de traiter tous les algorithmes et de simuler le comportement de n'importe quel ordinateur. Pour les plus simples, les cellules pourraient prendre trois valeurs et l'évolution de chaque cellule ne serait déterminée que par l'état de ses voisines immédiates. Celui présenté plus haut est de ce type. Une raison de plus pour le considérer avec respect.

Chapitre 4
STRUCTURES, MÉMOIRE, INFORMATION

Arrêtons-nous un instant pour jeter un regard en arrière et essayer de reconnaître les traits essentiels du domaine que nous venons de traverser.

Nous avons reconnu deux processus *a priori* différents :

– la résolution de problèmes par une "intelligence" ;
– le comportement adaptatif d'organismes vivants.

Or, et ce point est fondamental, la similitude des structures de ces deux processus est frappante. Dans les deux cas, nous trouvons :

– un "générateur de variété" qui, au cours du temps, choisit une solution ou un comportement parmi un ensemble de possibilités ;
– un "vérificateur", chargé de reconnaître si la solution proposée est une solution réelle qui satisfait les contraintes du problème ou, sous forme équivalente, si les variables essentielles de l'organisme restent à l'intérieur de leurs bornes physiologiques ;
– un "commutateur", qui bloque le générateur en cas de succès ou le relance dans le cas contraire.

De plus, l'un et l'autre processus peuvent, au moins dans certains cas, être simulés par des machines de structure analogue, tels l'homéostat. Le concept de base est alors celui de **stabilité** à l'intérieur du domaine défini par les contraintes du problème ou les limites des variables essentielles. La machine, activée par le générateur de variété, évolue d'une région à une autre jusqu'à la découverte d'un champ stable et s'y arrête.

Enfin, nous utilisons des ordinateurs, machines universelles, pour simuler les machines particulières qui viennent d'être évoquées.

L'une d'elles, l'homéostat, décrite en tant que prototype le plus simple, présente les caractéristiques suivantes :

– architecture à deux niveaux de contrôle ;

- boucles de rétroaction (feedback) au premier niveau, les champs correspondants pouvant être stables ou instables selon les valeurs des paramètres (matrice de couplage) ;
- recherche aveugle et aléatoire au deuxième niveau avec modification des paramètres, en cas d'instabilité ;
- système intégré dont les éléments sont richement interconnectés les uns aux autres sans que l'on puisse y distinguer des sous-ensembles.

Cette structure peut, nous l'avons vu, être efficace si le nombre des variables reste petit. Dans le cas contraire, les temps de recherche aveugle croissent de façon exponentielle avec la complexité, réduisant à zéro l'efficacité pratique. Nous l'allons montrer tout à l'heure. De plus, l'homéostat est amnésique. S'il découvre une adaptation correcte à un certain contexte et que celui-ci change, puis redevienne ce qu'il était, il sera impossible de retrouver facilement l'adaptation antérieure. La recherche repartira de zéro.

La création, dans le cadre de l'Intelligence artificielle, de processus plus efficaces de résolution de problèmes ou d'adaptation, utilisera, en particulier, deux stratégies sur lesquelles nous allons insister :

- tirer parti des connaissances relatives à la structure des problèmes (le sens précis du mot structure apparaîtra bientôt) ;
- mémoriser les solutions ou adaptations trouvées avec leurs contextes afin de constituer un stock d'expériences accumulées.

DU BON USAGE DES STRUCTURES

Limitations des systèmes intégrés

Prenons un exemple très simple.

N roues de loterie, analogues à celles utilisées dans les foires, comportent, chacune, trois secteurs égaux (ouverts à 120°) et caractérisés par A, B, C. Un bouton unique commande le mouvement des N roues. Chaque fois que l'opérateur appuie sur ce bouton, une machinerie fait tourner toutes les roues et les immobilise dans des positions aléatoires.

Lors de l'arrêt de chaque roue, chacun des trois secteurs a la même probabilité, soit 1/3, d'être situé en face d'un index. Le jeu consiste à relancer les roues jusqu'à ce que les N secteurs A soient tous en face des index.

Les variables essentielles, au nombre de N, peuvent prendre chacune trois valeurs : A, B, C. Les plages d'objectifs se réduisent à une valeur, soit A, pour chaque variable essentielle.

L'étude mathématique de ce jeu est simple. En voici les résultats principaux :

- à chaque lancement, la probabilité de succès est 1/3^N (soit 1 divisé par 3 puissance N). Ce nombre décroît très rapidement lorsque N augmente. Pour N=10 roues, il vaut 1/60000 environ. Pour N=20, il est inférieur au milliardième ;
- le nombre moyen d'épreuves nécessaires pour obtenir un succès est l'inverse de la probabilité précédente. Il vaut donc 3^N. Si chaque épreuve dure une minute, le temps moyen de recherche sera de 3^N minutes. Pour N=20, il atteindra 7000 ans.

Une illustration différente de la même structure serait offerte par un cambrioleur assez dépourvu de moyens et cherchant la combinaison d'un coffre au travers d'essais effectués au hasard. Si le coffre est équipé de N boutons, comportant chacun 10 chiffres, de 0 à 9, le nombre moyen d'essais accomplis par le cambrioleur néophyte, avant de trouver la combinaison, sera de 10^N. Pour N=5, cela donne 100000 et 10 milliards pour N=10.

Les exemples précédents, malgré leur simplicité, mettent en lumière la croissance exponentielle, donc extrêmement rapide, des durées de recherche aveugle, lorsque croît la complexité du système contrôlé. En termes biologiques, la solution d'un problème d'adaptation, même simple, aurait exigé, par cette méthode, des durées si considérables que jamais les organismes vivants n'auraient pu se développer sous les formes que nous connaissons.

Il est intéressant d'obtenir une formule plus générale pour l'indicateur de complexité que nous venons d'évoquer. Précisons-la. Définissons un système comprenant N variables essentielles qui peuvent, chacune, prendre Q valeurs. Le nombre d'états possibles du système, nous dirons sa "variété", est égal à Q^N. Supposons réalisées les conditions suivantes :

- une solution du problème (un état adapté) est associée à une combinaison de N valeurs particulières, prises par les N variables essentielles :
- une solution n'est reconnue que si *toutes* les conditions sont réalisées (toutes les variables essentielles ont pris les valeurs particulières qu'elles doivent atteindre) ;
- les moyens d'action ou commandes sont tels qu'une recherche aveugle et aléatoire, utilisant ces moyens, donne à toutes les valeurs de chaque variable essentielle une égale probabilité de se réaliser.

Dans ces conditions, le nombre moyen d'essais nécessaires pour trouver la solution est égal à la variété du système.

Systèmes séparables

Reprenons l'exemple du cambrioleur de coffres-forts. Supposons qu'il dispose d'instruments qui lui permettent, *pour chaque bouton,* de repérer le déclic lié au passage du bouton sur un chiffre de la combinaison. La situation est totalement modifiée. En dix essais, au maximum, le cambrioleur trouvera le chiffre associé à un bouton particulier. En 50 essais au plus (pour 5 boutons) et 100 (pour 10 boutons), il aura résolu son problème et ouvert le coffre. Rappelons les ordres de grandeur associés à la situation antérieure.

- 100000 essais pour 5 boutons ;
- 10 milliards d'essais pour 10 boutons.

Essayons de mieux comprendre les raisons de ces différences énormes. La situation initiale présente les caractéristiques suivantes :

- l'objectif (ouverture du coffre) n'est pas décomposable (en chiffres de la combinaison) ;
- les relations entre les commandes (positions des boutons) et la variable essentielle (coffre ouvert ou fermé) sont inconnues et non isolables. L'action d'un bouton particulier, quant à la réalisation de l'objectif, ne peut être distinguée. La seule possibilité est d'afficher, globalement, une configuration (positions données aux boutons) et d'essayer d'ouvrir le coffre. En cas d'insuccès, on recommence avec une autre configuration et ainsi de suite.

Dans la situation modifiée, au contraire :

- la variable essentielle globale (ouverture du coffre) est "éclatée" en ses composantes (chiffres affichés par chaque bouton) en nombre N égal au nombre de ces boutons ;
- les commandes sont, elles-mêmes, décomposées en N parties, constituées chacune par un bouton ;
- il y a une relation entre chacune des N commandes et une composante unique de la variable essentielle. Par contre, l'action d'une commande sur les autres composantes de l'objectif est nulle ;
- il est possible d'essayer chaque composante de la variable essentielle, c'est-à-dire de vérifier si un chiffre affiché par un bouton appartient, ou non, à la combinaison qui ouvre le coffre.

Un système qui présente ces dernières caractéristiques sera dit *"séparable"*. En raison de l'importance de cette notion, nous lui donnerons une forme précise, moyennant l'usage de quelques symboles que notre lecteur voudra bien excuser.

L'ensemble Y des variables essentielles est décomposable en N sous-ensembles Y_I. L'objectif global est atteint si les variables de chaque sous-ensemble restent à l'intérieur de leurs limites (éclatement des objectifs).

L'ensemble X des commandes est, également, décomposable en N sous-ensembles X_I. Si I est différent de J, les commandes contenues dans X_I n'ont aucune influence sur le variables essentielles de Y_J.

Le système initial (X, Y) est alors décomposable en sous-systèmes (X_I, Y_I) qui ne sont liés que par leurs variables essentielles (les Y_I, appartiennent à l'ensemble Y).

Un exemple classique de système séparable est constitué par un jeu d'équations linéaires dont chacune ne contient qu'une variable :

$$A_I * X_I = B_I$$

Les seconds membres B_I sont les objectifs éclatés. Chaque inconnue X_I (commande) peut être calculée à partir de la seule équation qui la contient. Un système linéaire général de la forme :

$$A_{11} * X_1 + \ldots \qquad + A_{1N} * X_N = B_1$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$A_{N1} * X_1 + \ldots. \qquad + A_{NN} * X_N = B_N$$

n'est évidemment pas séparable.

Comparons les indicateurs de complexité de deux systèmes particuliers.

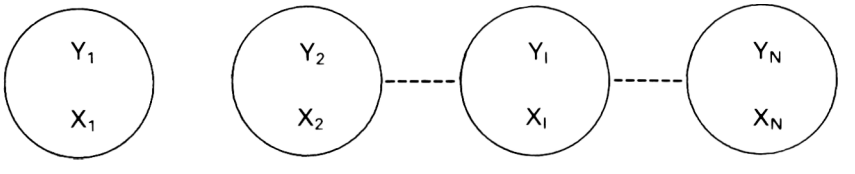
1er cas : système intégré

L'ensemble Y est formé de N variables essentielles Y_I qui peuvent, chacune, prendre P valeurs. D'où l'expression de la variété correspondante : V=P^N.

2e cas : système séparable

Chaque sous-système est constitué par une variable essentielle Y_I et les commandes qui lui sont liées. La variété correspondante est P. Pour le système global, la complexité et le temps de recherche sont proportionnels à P*N.

Dans le premier cas, la variété croît exponentiellement avec le nombre N des variables essentielles. Cette croissance est seulement linéaire dans le second cas. Dans un système intégré, les différentes parties sont richement interconnectées les unes aux autres. Un système séparable, au contraire, n'est qu'une juxtaposition de parties plus simples. Les relations d'influence entre les X_I et les Y_J dans un système séparable peuvent être, symboliquement, représentées comme suit (les pointillés dénotent des systèmes intermédiaires non figurés) :



Les roues de la fortune

A ce point de notre entretien, vous êtes, (ami)e) lecteur(trice), j'en ai le vif sentiment, quelque peu saturé(e) d'abstractions. De plus, votre Amstrad s'ennuie. Alors, un programme ?

Pourquoi ne pas simuler les roues de loterie qui, quelques pages plus haut, ont illustré la différence entre systèmes intégrés et séparables ? Voici donc le programme "LOTERIE". La copie d'écran ci-dessous indique ce qu'il propose.

```
$   $ $   $  LOTERIE   $   $ $   $

   ROUE 1      ROUE 2      ROUE 3
     A           A           A

 OPTIONS :

      0 --> Lancer les 3 roues
      1 --> lancer la roue 1
      2 --> lancer la roue 2
      3 --> lancer la roue 3

    <------ Votre choix ------>

    Nombre d'essais : 6
```

Vous êtes invité à :

- lancer simultanément les trois roues (option 0) ;
- lancer séparément chacune des roues 1,2,3 (options de même numéro).

Choisissons d'abord l'option 0 (système intégré). Les lettres A, B ou C sont tirées au sort (une par roue) et affichées. Recommençons jusqu'à obtenir trois "A", ce que, d'ailleurs, le programme signalera. Il vous indiquera également le nombre d'essais et le lot qui vous est généreu-

sement attribué. Tentez quelques expériences de ce premier type et enregistrez les nombres d'essais qui furent nécessaires à chaque fois. Tirez-en une statistique, la moyenne et la dispersion. La valeur théorique de cette moyenne est 3^3=27, mais la dispersion est très forte : les valeurs extrêmes peuvent varier de 1 à quelques centaines.

Essayez maintenant les jeux séparés. Itérez l'option 1 jusqu'à l'affichage de "A" pour la roue 1. Faites de même avec la roue 2 puis la roue 3. Notez le nombre total d'essais que le programme affichera. Refaites une statistique sur plusieurs parties et comparez. Ici, la moyenne calculée est de 3 pour chaque roue, soit 9 au total.

Il n'est pas nécessaire de commenter le programme. La logique en est simple. L'emploi des instructions WHILE...WEND et les remarques qui figurent dans le corps du programme le rendent assez clair. Voici donc les listes de variables et d'instructions.

Liste des variables

Tables :

LT$(3) : table contenant les lettres A, B, C.

Variables simples :

I : indice de boucle.
LT$: variable intermédiaire contenant une lettre.
N : nombre d'essais.
R : réponse à une question par un nombre.
R$: réponse à une question par oui ou non.
Z : nombre entier aléatoire.

Listing du programme LOTERIE

```
10 ' ------ INITIALISATION
20 RANDOMIZE TIME
30 MODE 1:PAPER 13:PEN 0:CLS
40 WINDOW#1,7,14,6,10:PAPER#1,0:CLS#1
50 WINDOW#1,6,13,5,9:PAPER#1,10:PEN#1,0:
CLS#1
60 WINDOW#2,18,25,6,10:PAPER#2,0:CLS#2
70 WINDOW#2,17,24,5,9:PAPER#2,10:PEN#2,0
:CLS#2
80 WINDOW#3,29,36,6,10:PAPER#3,0:CLS#3
90 WINDOW#3,28,35,5,9:PAPER#3,10:PEN#3,0
:CLS#3
100 WINDOW#4,4,38,14,22:PAPER#4,0:CLS#4
```

```
110 WINDOW#4,3,37,13,21:PAPER#4,10:PEN#4
,0:CLS#4
120 FOR i=1 TO 15
130 LOCATE i,2:PEN (i MOD 10):PRINT "$"
140 LOCATE 41-i,2:PRINT "$"
150 NEXT
160 LOCATE 17,2:PEN 0:PRINT "LOTERIE"
170 FOR i=1 TO 3
180 LOCATE#i,2,2:PRINT#i,"ROUE";i
190 NEXT
200 LOCATE 8,24:PRINT "Nombre d'essais :
"
210 ' ------- MENU
220 LOCATE#4,3,2:PRINT#4,"OPTIONS :"
230 LOCATE#4,8,4:PRINT#4,"0 --> Lancer l
es 3 roues"
240 FOR i=1 TO 3
250 LOCATE#4,7,4+i:PRINT#4,i;"--> lancer
 la roue";i
260 NEXT
270 LOCATE#4,6,9:PRINT#4,"<------- Votre
choix -------->"
280 ' ---- BOUCLE PRINCIPALE
290 WHILE NOT(lt$(1)="A" AND lt$(2)="A"
AND lt$(3)="A")
300 ' ---- LANCEMENT DES ROUES
310 GOSUB 850
320 ON r+1 GOSUB 580,670,730,790
330 '
340 ' 580-> toutes les roues
350 ' 670,730,790 -> 1ere, 2eme et 3eme
roue
360 '
370 ' ---- RESULTATS
380 n=n+1
390 LOCATE 25,24:PRINT STRING$(4," ")
400 LOCATE 25,24:PRINT n
410 WEND
420 ' ----- LA FETE
430 ENT 2,=67,24,=86,20,=127,24,=156,20
440 SOUND 1,0,100,15,,2
450 CLS#4:LOCATE#4,2,5:PRINT#4,"Vous gag
nez";INT(9/n);"raton(s) laveur(s)"
460 ' ----- SORTIE OU REPRISE
470 LOCATE#4,2,8:PRINT#4,"Voulez-vous re
jouer ( o ou n ) ? ";
480 r$=""
490 WHILE r$<>"o" AND r$<>"n"
500   r$=LOWER$(INKEY$)
```

```
510 WEND
520 PRINT#4,r$
530 IF r$="o" THEN CLEAR:RUN
540 CLS:END
550 '
560 '     TOUTES LES ROUES
570 '
580 FOR i=1 TO 3
590   GOSUB 940:' tirage
600   lt$(i)=lt$
610   LOCATE#i,4,4:PRINT#i,lt$
620 NEXT i
630 RETURN
640 '
650 '     1ERE ROUE
660 '
670 GOSUB 940:lt$(1)=lt$
680 LOCATE#1,4,4:PRINT#1,lt$
690 RETURN
700 '
710 '     2EME ROUE
720 '
730 GOSUB 940:lt$(2)=lt$
740 LOCATE#2,4,4:PRINT#2,lt$
750 RETURN
760 '
770 '     3EME ROUE
780 '
790 GOSUB 940:lt$(3)=lt$
800 LOCATE#3,4,4:PRINT#3,lt$
810 RETURN
820 '
830 '     CHOIX
840 '
850 r$=""
860 WHILE r$<"0" OR r$>"9"
870   r$=INKEY$
880 WEND
890 r=VAL(r$)
900 RETURN
910 '
920 '     TIRAGE
930 '
940 z=INT(3*RND+1)
950 lt$=CHR$(64+z)
960 SOUND 1,255,10
970 RETURN
```

Systèmes ordonnés

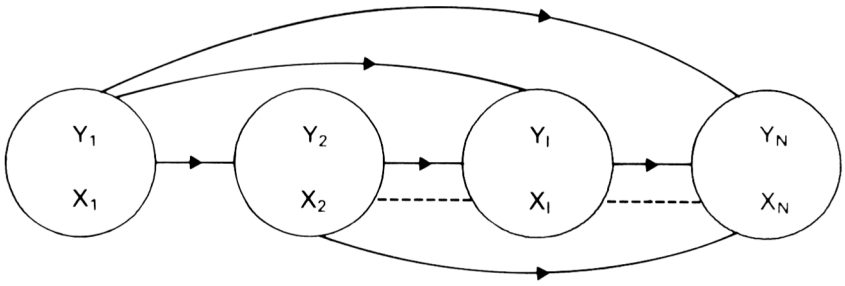
La facilité de contrôle, constatée pour les systèmes séparables, s'étend à une autre catégorie : celle des systèmes *ordonnés.*

Pour ceux-ci :

- l'ensemble Y des variables essentielles est décomposable en N sous-ensembles Y_1, Y_2.....Y_N, pris dans cet ordre ;
- l'ensemble X des commandes se divise en N sous-ensembles X_1, X_2.....X_N pris dans cet ordre ;
- les relations entre décisions et résultats sont telles que :

  seul X_1 peut agir sur Y_1,
  seuls X_1 et X_2 peuvent agir sur Y_2,
  et, en général, seuls les X_J de rang $J<=I$ peuvent agir sur Y_I.

La figure suivante symbolise ces relations :



Le degré de complexité du système global est alors comparable à celui d'une de ses N parties. En effet, les commandes X_1 pourront être recherchées en tenant compte des seuls objectifs associés à Y_1 et des seules relations entre les ensembles X_1 et Y_1. Les X_1 étant ainsi définies, le problème de détermination des X_2 met en jeu les seules variables du sous-système (X_2, Y_2) et ainsi de suite.

Les équations linéaires générales évoquées plus haut peuvent être résolues précisément parce que transformables en systèmes ordonnés et même séparables. Les opérations correspondantes portent le nom de triangulation (système ordonné) et de diagonalisation (système séparable) de la matrice.

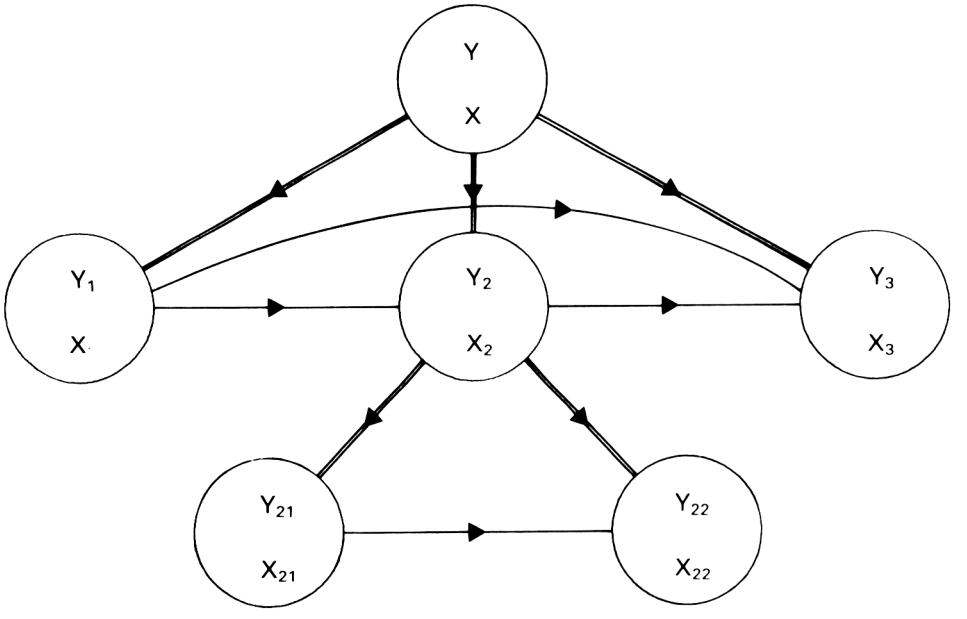
Systèmes hiérarchisés

L'opération qui décompose en sous-systèmes un système séparable ou ordonné peut, dans certains cas, être itérée, c'est-à-dire recommencée. Le système sera alors dit *hiérarchisé.* Un au moins des sous-

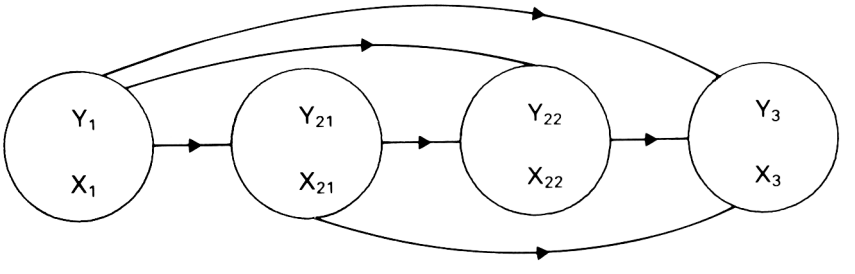
systèmes, soit (X_I, Y_I) sera alors séparable ou ordonné. On pourra ainsi définir :

- plusieurs sous-ensembles de variables essentielles Y_{IK} ;
- plusieurs sous-ensembles de commande X_{IK}.

Les X_{IK} agiront sur les Y_{IK} selon les modes précisés plus haut. Les opérations pourront être répétées sur plusieurs *niveaux.* Voici un exemple, représenté symboliquement, de système (X,Y), éclaté sur trois niveaux :



Les arcs simples symbolisent les interactions entre sous-systèmes. Les arcs doubles représentent l'éclatement des objectifs et la séparation en sous-systèmes composants. Au terme des opérations, le système (X, Y) sera remplacé, aux fins de contrôle, par le réseau ordonné suivant :



Nous comprenons ainsi comment il est possible de contrôler, en un temps raisonnable, un système complexe, sous la condition qu'il soit hiérarchisé.

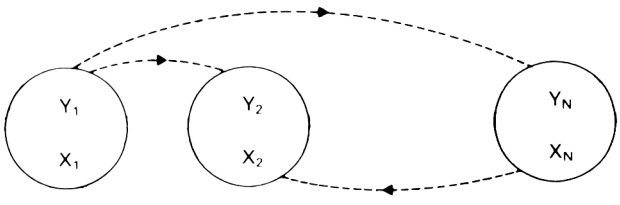
Systèmes multistables et presque hiérarchisés

Les problèmes réels que doit résoudre l'intelligence humaine se rattachent à des systèmes qui ne sont que rarement séparables ou ordonnés au sens strict. Le point essentiel est qu'ils soient *presque* séparables ou ordonnés ou encore qu'ils puissent être ramenés au voisinage de telles formes.

Il est rare, en effet, que chaque commande agisse, sans distinction, sur toutes les variables essentielles. Une telle richesse de couplages et d'interconnexions est exceptionnelle. Lorsqu'elle se rencontre, le contrôle est pratiquement impossible. Mais, dans tous les autres cas, le système va présenter une *structure* qu'il sera capital de reconnaître. L'exemple le plus simple d'une telle structure consiste en une décomposition du système (X, Y) en sous-systèmes (X_I, Y_I) tels que :

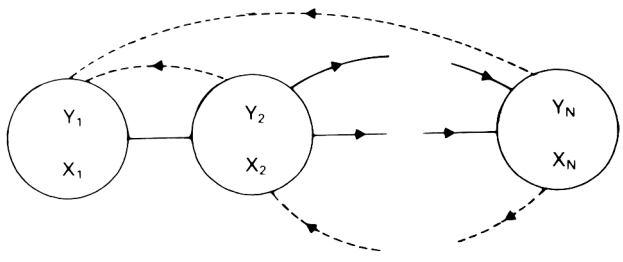
- les liaisons entre les variables de X_I et celles de Y_I soient multiples, serrées et fréquentes ;
- les connexions entre deux sous-systèmes (X_I, Y_I) et (X_J, Y_J) soient peu nombreuses et rarement actives.

Ross Ashby (*cf. Design for a brain,* déja cité, chap. 16) a nommé *multistables* de tels systèmes qui sont presque séparables. Leur représentation symbolique pourrait être la suivante.



Les liaisons en pointillé représentent des connexions rares et peu actives entre les sous-systèmes.

De la même manière, un système presque ordonné pourrait être ainsi symbolisé :



Ici encore, les arcs en pointillé représentent des connexions rares et peu actives qui interviennent en sens inverse des couplages normaux conformes à la définition d'un système ordonné.

De façon plus générale, nous appellerons *presque hiérarchisé* un système qui est hiérarchisé si l'on excepte certaines liaisons peu nombreuses et rarement actives entre les sous-systèmes qui le constituent. Les systèmes de ce type présentent un intérêt majeur. Pour résoudre les problèmes qui les concernent, l'intelligence humaine utilise une stratégie générale dont les grandes lignes apparaissent ainsi :

- isoler des sous-systèmes semi-autonomes, les "holons" (*cf.* Arthur Koestler, *Le cheval dans la locomotive,* Calmann-Lévy Editeur) à partir de toutes les informations disponibles quant à la structure ;
- imaginer et définir, pour chacun d'eux, des objectifs aussi cohérents que possible avec ceux du système global ;
- déterminer, pour chacun d'eux, à partir des objectifs locaux, les commandes ou décisions optimales, comme si le système total était parfaitement hiérarchisé ;
- corriger, au cours du temps, les dérives et distorsions dues aux interactions non prises en compte.

La mise en œuvre de stratégies de ce type relève de l'heuristique. Le succès, comme l'échec, par conséquent, sont toujours provisoires.

Une classe particulière de systèmes presque hiérarchisés joue un rôle important dans le domaine de l'Intelligence artificielle : les réseaux (arbres ou graphes) dans lesquels chaque décision conditionne les suivantes. Les jeux à une ou deux personnes, de hasard ou de réflexion, en sont des exemples connus. La stratégie générale s'applique. Les objectifs locaux s'appellent ici fonction d'évaluation. Les sous-systèmes considérés sont définis par le nombre de niveaux simultanément pris en compte dans l'arbre ou le graphe. Le calcul des solutions utilise la structure particulière de ces problèmes. Nous le vîmes au chapitre 1, dans le cas de jeux à une personne (Taquin). Nous l'allons revoir en détail dans la deuxième partie de ce livre, consacrée aux jeux de réflexion à deux personnes.

MÉMOIRE ET APPRENTISSAGE

L'une des grandes faiblesses de l'homéostat et des machines de même type est, nous l'avons vu, leur amnésie. Elles ne retiennent pas les solutions trouvées à des problèmes déjà rencontrés. Elles ne peuvent reconnaître une situation ancienne, rechercher dans leur mémoire le comportement adapté et l'utiliser immédiatement.

Pouvons-nous simuler, sur ordinateur, une "machine" qui apprenne et se souvienne ? Entendons-nous bien. Ce dispositif doit être d'usage général et utilisable à toutes fins. Quelles doivent être ses performances ? Le problème à résoudre est défini par deux listes et leurs liaisons.

La première liste se réfère aux critères ou conditions qui peuvent caractériser une situation définie. S'il s'agit, par exemple, de décrire la rencontre d'un animal, ces conditions pourraient être :

- il a une crinière,
- il a des plumes,
- son pelage est rayé,
- il est ongulé,
- il vole,
- etc.

Pour une maladie, ce sera la présence ou l'absence de symptômes. Chaque condition doit être posée comme vraie ou non (binaire). Toute situation est représentée par une sous-liste de conditions vraies, simultanément réalisées.

La seconde liste est celle des réponses pour chaque situation définie. Ce peut être l'indication de la classe à laquelle appartient cette situation (espèce d'un animal, nom d'une maladie, etc.) ou le comportement à observer (s'enfuir, se cacher pour un animal, prescrire un traitement pour une maladie...).

De plus la machine devra apprendre. Elle est, au départ, vierge de toute idée préconçue. Chaque situation lui est présentée dans un ordre quelconque et décrite par la donnée des conditions vérifiées. La machine "devine" une réponse et la propose. Si la réponse est correcte, une autre situation est appelée, sinon, la réponse adéquate est indiquée. Après un certain nombre d'expériences de ce type, la machine doit être capable de fournir, pour chaque cas, la réponse convenable. Notre tâche est d'écrire un programme qui réponde à ce cahier des charges. C'est fait. Mais avant d'en décrire la structure interne, mettons-le à l'épreuve.

La vie difficile d'un oisillon

Le programme est général. Nous pourrions donc, et vous y êtes cordialement invité, choisir des exemples sérieux tels que :

- les villes de France caractérisées par la région et l'importance de leur population ;
- les tables de multiplication ;
- le choix de conditions financières pour un investissement d'entreprise ou familial ;
- les conjugaisons des verbes français ;

– les déclinaisons latines ;
– les pannes d'une voiture ;
– etc.

J'ai préféré, sur les traces de Konrad Lorentz, extraire de mes longues et patientes observations les détails de l'éducation d'un petit oiseau et de l'expérience, chèrement acquise, transmise par ses parents. Lançons le programme.

Premier stade : il demande la liste des conditions ou critères rencontrés. La copie d'écran ci-dessous les indique.

```
Entrez les criteres qui doivent conditionner mes reponses.

Nombre de criteres? 7

Critere No. 1 ? A des feuilles
Critere No. 2 ? Vole
Critere No. 3 ? Est tout petit
Critere No. 4 ? A quatre pattes
Critere No. 5 ? A des moustaches
Critere No. 6 ? A de grandes ailes
Critere No. 7 ? Rampe
```

Viennent ensuite les réponses possibles. Les voici.

```
Entrez les reponses possibles.

Nombre de reponses? 6

Reponse No. 1 ? Me percher
Reponse No. 2 ? L'attraper et le manger
Reponse No. 3 ? Rester vigilant
Reponse No. 4 ? M'enfuir a tire d'aile
Reponse No. 5 ? Me cacher
Reponse No. 6 ? Defendre mes oeufs
```

L'expérience enseignée regroupe les préceptes suivantes :

SI "A des feuilles" ALORS "Me percher"
SI "Vole et est tout petit " ALORS "L'attaper et le manger"
SI "A quatre pattes" ALORS "Rester vigilant"

SI "A quatre pattes" ET "A des moustaches" ALORS "M'enfuir à tire d'ailes"
SI "Vole" ET "A de grandes ailes" ALORS "Me cacher"
SI "Rampe" ALORS "Défendre mes œufs".

Sur demande du programme, nous allons donc lui présenter, autant de fois que nécessaire, chacune de ces situations, en précisant les conditions vérifiées. Le programme réfléchit, propose sa réponse (première copie d'écran qui suit). Si elle convient, nous l'indiquons et continuons. Sinon, nous le disons aussi et le programme demande quelle aurait été la réponse correcte (deuxième copie d'écran).

```
                    MEMOIRE

 Je ne demande qu'a apprendre. Pour chaque situation envisagee
indiquez moi, dans la liste ci-dessous des criteres
ceux qui sont verifies (reponse o) ou non (reponse n).
 Je proposerai une reponse et vous m'indiquerez si elle est correcte.


    A des feuilles ?                                  n
    Vole ?                                            o
    Est tout petit ?                                  o
    A quatre pattes ?                                 n
    A des moustaches ?                                n
    A de grandes ailes ?                              n
    Rampe ?                                           n

  Reponse 'Defendre mes oeufs'  Est-elle correcte ?
```

```
                    MEMOIRE

 Je ne demande qu'a apprendre. Pour chaque situation envisagee
indiquez moi, dans la liste ci-dessous des criteres
ceux qui sont verifies (reponse o) ou non (reponse n).
 Je proposerai une reponse et vous m'indiquerez si elle est correcte.
```

```
Voici la liste des reponses.
Indiquez le numero de la reponse correcte.

 1 :- Me percher
 2 :- L'attraper et le manger
 3 :- Rester vigilant
 4 :- M'enfuir a tire d'ailes
 5 :- Me cacher
 6 :- Defendre mes oeufs

 Numero correct?
```

Après deux ou trois présentations de chaque cas, le programme triomphe et, dès cet instant, toutes ses réponses seront bonnes. Son éducation est achevée.

```
                          MEMOIRE

 Je ne demande qu'a apprendre. Pour chaque situation envisagee
indiquez moi, dans la liste ci-dessous des criteres
ceux qui sont verifies (reponse o) ou non (reponse n).
 Je proposerai une reponse et vous m'indiquerez si elle est correcte.

    A des feuilles ?                                  n
    Vole ?                                            n
    Est tout petit ?                                  n
    A quatre pattes ?                                 o
    A des moustaches ?                                o
    A de grandes ailes ?                              n
    Rampe ?                                           n

 Reponse 'M'enfuir a tire d'ailes'  Est-elle correcte ?o
Je crois avoir compris !
 Nous continuons ?
```

Faisons maintenant un tour à la cuisine.

La mémoire dans la puce

Au chapitre 1, fut présenté un premier modèle d'apprentissage (programme "CHIFFRES") basé sur l'enregistrement des erreurs effectuées qui ne doivent pas se répéter. Le modèle ici proposé est plus général. L'idée centrale vient de la théorie statistique de la "discrimination" ou de la "classification".

Elle associe à chaque réponse (nombre des réponses=NR) un vecteur progressivement construit au cours des essais. Chaque vecteur est formé d'entiers, positifs ou négatifs, en nombre égal au nombre NC des critères. Si l'on place

- en lignes les NC critères, d'indices I,
- en colonnes les NR réponses, d'indices J,

les vecteurs-colonnes forment une matrice de corrélation entre critères et réponses, matrice représentée par la table COR(I,J).

Soit VLC(I) les valeurs logiques (1 si OUI, 0 si NON) associées aux divers critères CR$(I), d'indices I, lors d'un essai. A chaque réponse RE$(J), d'indice J, est associé un rang RANG(J) calculé en multipliant les valeurs logiques VLC(I) par les éléments homologues de la colonne J de la table COR(I,J) et sommant les produits correspondants. Pour ceux qui ne craignent pas les notions mathématiques, le vecteur RANG est le produit du vecteur VLC par la matrice COR. Le programme propose la réponse, d'indice JP, dont le rang est maximal (le dernier maximum rencontré en cas d'égalité). Ce processus est traduit dans les instructions 180 à 380 du programme "MEMOIRE".

```
180 ' -------- Apprentissage
190 CLS #1
200 PRINT " Je ne demande qu'a apprendre
. Pour chaque situation envisagee"
210 PRINT "indiquez moi, dans la liste c
i-dessous des criteres"
220 PRINT "ceux qui sont verifies (repon
se o) ou non (reponse n)."
230 PRINT " Je proposerai une reponse et
 vous m'indiquerez si elle est correcte.
"
240 maxr=0
250 FOR i=1 TO nc
260   PRINT #1,TAB(5);cr$(i);" ? ";TAB(6
0):GOSUB 630:'Oui ou non
270   IF LOWER$(r$)="o" THEN vlc(i)=1 EL
SE vlc(i)=0
280 NEXT i
290 FOR j=1 TO nr
300   rang(j)=0
```

```
310    FOR i=1 TO nc
320      rang(j)=rang(j)+vlc(i)*cor(i,j)
330    NEXT i
340    IF rang(j)>=maxr THEN maxr=rang(j)
:jp=j
350 NEXT j
360 PRINT #1:PRINT #1," Reponse '";re$(j
p);"'  Est-elle correcte ?";:GOSUB 630:'
Oui ou non
370 ' -------- Reponse correcte
380 IF LOWER$(r$)="o" THEN vr(jp)=1:GOTO
 550
```

Le point important est le mode de calcul, relativement simple, de la table COR(I,J). Au départ, tous les éléments sont initialisés à zéro. En cas de succès, la table reste inchangée. En cas d'échec, l'indice JC de la réponse correcte est noté. Puis le programme modifie, comme suit, en les balayant, les colonnes d'indice J de la table COR(I,J) :

- pour toute colonne d'indice J différent de JC et dont le rang est supérieur ou égal au maximum trouvé, les valeurs logiques VLC(I) des critères sont soustraites des éléments de même indice de la table ;
- les mêmes valeurs logiques VLC(I) sont ajoutées aux éléments d'indice I de la colonne JC contenant la réponse correcte.

Ainsi, la réponse correcte est favorisée et ses concurrentes pénalisées. Les chances d'obtenir la réponse correcte au prochain essai sont ainsi accrues. Voici les instructions correspondantes (lignes 390 à 530) du programme.

```
390 ' -------- Reponse erronee
400 CLS #1
410 PRINT #1,"Voici la liste des reponse
s."
420 PRINT #1,"Indiquez le numero de la r
eponse correcte.":PRINT #1
430 FOR j=1 TO nr
440   vr(j)=0
450   PRINT #1,j;":- ";re$(j)
460 NEXT j:PRINT #1
470 INPUT #1," Numero correct";jc
480 FOR j=1 TO nr
490   IF rang(j)>=maxr AND j<>jc THEN FO
R i=1 TO nc:cor(i,j)=cor(i,j)-vlc(i):NEX
T i
500 NEXT j
```

```
510 FOR i=1 TO nc
520    cor(i,jc)=cor(i,jc)+vlc(i)
530 NEXT i
```

A la fin des opérations et pour l'exemple choisi, la table COR(I,J) offrira, par exemple, l'aspect suivant.

1	-1	-1	-1	-1	-1
-1	0	-1	-1	0	-1
-1	1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1
0	0	-1	1	0	0
0	-1	0	0	1	0
0	0	0	0	0	0

Le programme "MÉMOIRE"

Ce programme est un "système expert", certes simple, mais assez général. Il traite tout groupe, extérieur au programme, de relations logiques du type :

Si cond. 1 ET cond. 2....ET cond.N ALORS réponse X

lorsque conditions et réponses forment des ensembles disjoints. Il constitue, à ce titre, une introduction aux méthodes et programmes développés dans la troisième partie de ce livre, partie précisément consacrée aux systèmes experts. Le voici, précédé d'une liste complète des variables utilisées.

Liste des variables

Tables :

COR(NC,NR) : table de corrélation entre critères et réponses.
CR$(NC) : critères ou conditions en nombre NC.
RANG(NR) : rang de classement des réponses.
RE$(NR) : réponses en nombre NR.
VLC(NC) : valeurs logiques (1 ou 0) attribuées à chaque critère.
VR(NR) : valeur de succès des réponses (1 pour bonne réponse, 0 sinon).

Variables simples :

I,J : indices de boucle.
JC : numéro de la réponse correcte.

JP : numéro de la réponse proposée.
MAXR : maximum du rang de classement des réponses.
NC : nombre de critères ou conditions.
NR : nombre de réponses.
R$: réponse par Oui ou Non.
SCORE : nombre de bonnes réponses depuis la dernière erreur.

Listing du programme "MÉMOIRE"

```
10 ' -------- Initialisation
20 INK 0,1:INK 1,24:PAPER 0:PEN 1
30 MODE 2:CLS
40 DEFINT a-z
50 PRINT TAB(36)"MEMOIRE":PRINT
60 WINDOW #1,2,79,9,24
70 ' -------- Entree des criteres et des
reponses
80 CLS #1:PRINT #1,"Entrez les criteres
qui doivent conditionner mes reponses.":
PRINT #1
90 INPUT #1,"Nombre de criteres";nc:DIM
cr$(nc),vlc(nc):PRINT #1
100 FOR i=1 TO nc
110   PRINT #1,"Critere No.";i;:INPUT #1
,cr$(i)
120 NEXT i
130 CLS #1:PRINT #1,"Entrez les reponses
 possibles.":PRINT #1
140 INPUT #1,"Nombre de reponses";nr:DIM
 re$(nr),rang(nr),vr(nr),cor(nc,nr):PRIN
T #1
150 FOR i=1 TO nr
160   PRINT #1,"Reponse No.";i;:INPUT #1
,re$(i)
170 NEXT i
180 ' -------- Apprentissage
190 CLS #1
200 PRINT " Je ne demande qu'a apprendre
. Pour chaque situation envisagee"
210 PRINT "indiquez moi, dans la liste c
i-dessous des criteres"
220 PRINT "ceux qui sont verifies (repon
se o) ou non (reponse n)."
230 PRINT " Je proposerai une reponse et
 vous m'indiquerez si elle est correcte.
"
240 maxr=0
```

```
250 FOR i=1 TO nc
260   PRINT #1,TAB(5);cr$(i);" ? ";TAB(6
0):GOSUB 630:'Oui ou non
270   IF LOWER$(r$)="o" THEN vlc(i)=1 EL
SE vlc(i)=0
280 NEXT i
290 FOR j=1 TO nr
300   rang(j)=0
310   FOR i=1 TO nc
320     rang(j)=rang(j)+vlc(i)*cor(i,j)
330   NEXT i
340   IF rang(j)>=maxr THEN maxr=rang(j)
:jp=j
350 NEXT j
360 PRINT #1:PRINT #1," Reponse '";re$(j
p);"'  Est-elle correcte ?";:GOSUB 630:'
Oui ou non
370 ' -------- Reponse correcte
380 IF LOWER$(r$)="o" THEN vr(jp)=1:GOTO
 550
390 ' -------- Reponse erronee
400 CLS #1
410 PRINT #1,"Voici la liste des reponse
s."
420 PRINT #1,"Indiquez le numero de la r
eponse correcte.":PRINT #1
430 FOR j=1 TO nr
440   vr(j)=0
450   PRINT #1,j;":- ";re$(j)
460 NEXT j:PRINT #1
470 INPUT #1," Numero correct";jc
480 FOR j=1 TO nr
490   IF rang(j)>=maxr AND j<>jc THEN FO
R i=1 TO nc:cor(i,j)=cor(i,j)-vlc(i):NEX
T i
500 NEXT j
510 FOR i=1 TO nc
520   cor(i,jc)=cor(i,jc)+vlc(i)
530 NEXT i
540 ' -------- Test de fin d'apprentissag
e
550 score=0
560 FOR j=1 TO nr
570   score=score+vr(j)
580 NEXT j
590 IF score=nr THEN PRINT #1,"Je crois
avoir compris !"
600 ' -------- Sortie du programme ou rep
rise
```

```
610 PRINT #1," Nous continuons ?";:GOSUB
 630:'Oui ou non
620 IF LOWER$(r$)="o" THEN CLS #1:GOTO 2
40 ELSE END
630 '
640 '        OUI OU NON
650 '
660 r$=INKEY$
670 IF r$="" OR (LOWER$(r$)<>"o" AND LOW
ER$(r$)<>"n") THEN 660
680 PRINT #1,r$
690 RETURN
```

RECHERCHE ET INFORMATION

Au cours des pages précédentes, nous avons rencontré des problèmes de difficultés variées et constaté les différences d'efficacité des techniques employées pour les résoudre. Par exemple, la rétroaction et la reconnaissance de structures séparables, ordonnées ou hiérarchisées constituent des méthodes puissantes et relativement générales.

Il apparaît ainsi intéressant de définir des concepts qui pourraient permettre de **mesurer** la difficulté globale d'un problème, de **calculer,** *a priori* et *a posteriori,* les efficacités des diverses méthodes imaginables de recherche, afin de guider les choix.

Une telle notion existe. Elle fut développée par Shannon et Weaver en 1949 et forme la base d'une théorie mathématique de l'information (*cf. The mathematical Theory of communication* par Claude E. Shannon et Warren Weaver, The University of Illinois Press, Urbana, 1949).

Cette théorie joue un rôle central dans les problèmes qui nous occupent. Montrons-le sur quelques exemples simples.

Le problème du dictionnaire

Deux joueurs sont en présence. Le premier choisit, au hasard, un mot de la langue française. Le second doit trouver ce mot et, pour cela, pose des questions à réponse binaire (par oui ou non).

La difficulté du problème peut être caractérisée par un choix parmi N mots également probables. Ici N est voisin de :

$$2^{15} = 32768$$

Pour plusieurs raisons, évoquées plus loin, la **quantité d'incertitude** attachée au problème est évaluée par le logarithme de base 2 du

nombre N des choix possibles également probables et s'exprime en bits. Ici :

$$\log2(2\hat{}15)=15 \text{ bits}$$

Le choix d'une mesure logarithmique est dicté par plusieurs considérations :

a) Le nombre des supports techniques de l'information (relais, transistors, etc.) varie linéairement avec le logarithme du nombre des possibilités. Par exemple, l'addition d'un relais à un groupe de relais double le nombre des états possibles du système ainsi constitué, ce qui revient à ajouter 1 au logarithme de base 2 de ce nombre ;

b) Nous pensons intuitivement que deux cartes perforées ont une capacité d'information double de celle correspondant à une seule carte. Or le nombre des états possibles est élevé au carré ;

c) Notre troisième remarque est importante : le gain d'information obtenu en passant d'une configuration de N1 états à une autre configuration de N2 états doit être **invariant** si l'on change les unités de mesure ou le système de référence (cela est nécessaire si la quantité d'information doit correspondre à une réalité physique). La fonction log(N) a précisément cette propriété. Le gain d'information ou la réduction d'incertitude s'expriment ainsi par :

$$\log2(N1)-\log2(N2)=\log2(N1/N2)$$

Le rapport N1/N2 est indépendant des unités.

Les mêmes raisons conduisent à généraliser, sous la forme suivante, la mesure de l'incertitude. Soit N états possibles E_i affectés chacun de probabilités P_i. La quantité d'incertitude, relative au choix d'un état particulier dans cet ensemble a pour expression :

$$I=-\Sigma p_i*\log2(p_i)$$

Si les divers états sont également probables, toutes les probabilités sont égales à 1/N et l'on retrouve :

$$I=-N*(1/N)*\log2(1/N)=\log2(N)$$

Appliquons ces notions à l'étude des efficacités comparées de trois méthodes qui peuvent être employées par le joueur B pour trouver le mot choisi par le joueur A.

Sélection d'un mot au hasard

B choisit, au hasard, un mot français et pose la question : ce mot est-il le bon ? La réponse est oui ou non. Demandons-nous quelle est la quantité d'information contenue dans la réponse à une telle question. Deux possibilités existent :

a) Réponse Oui : le joueur B a trouvé du premier coup. La probabilité de cet événement est :

$$1/N=1/(2\hat{}15)=1/32768$$

Le gain d'information est égal à log2(N), soit 15 bits, puisque le joueur B passe d'une incertitude évaluée par log2(N) à une certitude (incertitude nulle).

b) Réponse Non : la probabilité correspondante est le complément à 1 de la précédente, soit 1−1/N. La nouvelle incertitude correspond à un ensemble de N−1 choix (l'une des possibilités a été éliminée) et s'exprime par log2(N−1). Le gain d'information est égal à la réduction d'incertitude, soit : log2(N)−log2(N−1)=log2(N/N−1).

L'évaluation *"a priori"* (c'est-à-dire avant que la réponse soit connue) du gain d'information G, fourni par la première question, s'obtient en pondérant les estimations relatives à chaque réponse (oui et non) par leurs probabilités respectives. Cela s'appelle calculer l'espérance mathématique. On trouve ainsi :

$$G=1/N*\log2(N)+(1-1/N)*\log2(N/N-1)$$

soit, tous calculs faits : 0,0005 bits.

En conclusion, la quantité d'information gagnée par la première question vaut environ 1/2 millième de bit. Elle est donc extrêmement faible. Certes, l'information espérée de chaque question va croître lentement avec le nombre des questions posées. Chaque mot éliminé a, en effet, plus d'importance à mesure que diminue le nombre des mots restants. A la limite, si tous les mots du dictionnaire, sauf deux, ont été successivement écartés, l'information espérée de la dernière question vaudra 1 bit. L'information moyenne, attendue de chaque question au cours du processus, sera ainsi voisine du millième de bit. Pour recueillir les 15 bits d'information exigés par la solution du problème (trouver le mot inconnu), il faudra donc 15 000 questions en ordre de grandeur. La méthode utilisée apparaît ainsi très peu efficace et, sur ce point, l'analyse confirme l'intuition.

Le programme SHANNON

Un programme serait bien agréable pour effectuer à notre place ces divers calculs. Nous l'avons baptisé SHANNON. Voici, pour l'exemple précédent et en copie d'écran, le type d'informations qu'il produit.

```
                    SHANNON

Nombre total d'articles? 32768
Numero d'ordre d'un essai? 5

        RESULTATS

Valeur (en bits) du gain d'information a obtenir: 15
```

```
Probabilite d'insucces pendant 4 essai(s): 0.99988
Gain d'information associe: 0.00018

Probabilite de succes a l'essai no. 5 : 0.00003
Gain d'information associe: 14.99982

Probabilite d'insucces a l'essai no. 5 : 0.99997
Gain d'information associe: 0.00004

Gain espere d'information de l'essai no. 5 apres 4 echec(s): 0.0005
Rang moyen de l'essai qui reussit: 16384.5
Gain moyen d'information par essai: 0.00092

Voulez vous continuer (o ou n) ?n
```

Quelques remarques :

- le nombre total N d'éléments de l'ensemble (ici des mots) et le rang ou numéro d'ordre I de la question posée peuvent être choisis arbitrairement ; faites varier I pour observer l'évolution du gain espéré d'information pour chaque question ;
- si le succès est obtenu à la Ie question, la somme des quantités d'information recueillies au cours des (I−1) échecs précédents et de celle associée au succès doit égaler log2(N). Pour N=15 et I=5 (exemple précédent), on vérifie que 0,00018+14,99982=15 ;
- le rang moyen de l'essai qui réussit est (N+1)/2.

Voici les listes de variables et d'instructions.

Liste des variables

D	: nombre de décimales.
G	: espérance mathématique du gain d'information au Ie essai.
I	: numéro d'ordre d'une question (ou d'un essai).
N	: nombre d'éléments ou d'articles dans l'ensemble de recherche.
N1, N2	: variables intermédiaires.
PINS	: probabilité d'insuccès au Ie essai.
PS	: probabilité de succès au Ie essai.
PTINS	: probabilité cumulée des insuccès avant le Ie essai.
RG	: rang moyen de l'essai qui réussit.

R$: réponse par oui ou non.
Z : variable intermédiaire.

Listing du programme SHANNON

```
10 INK 0,1:INK 1,24:PAPER 0:PEN 1
20 MODE 2
30 CLS:PRINT TAB(36)"SHANNON":PRINT
40 d=5
50 INPUT" Nombre total d'articles";n
60 INPUT" Numero d'ordre d'un essai";i
70 n1=n-i+1:n2=n-i
80 PRINT:PRINT TAB(10)"RESULTATS":PRINT
90 PRINT " Valeur (en bits) du gain d'in
formation a obtenir:";:z=LOG(n)/LOG(2):G
OSUB 290
100 ptins=1-(i-1)/n:ps=1/(n-(i-1)):pins=
1-ps
110 PRINT:PRINT " Probabilite d'insucces
 pendant";i-1;"essai(s):";:z=ptins:GOSUB
 290
120 PRINT " Gain d'information associe:"
;:z=LOG(n/n1)/LOG(2):GOSUB 290
130 PRINT:PRINT " Probabilite de succes
a l'essai no.";i;":";:z=ps:GOSUB 290
140 PRINT " Gain d'information associe:"
;:z=LOG(n1)/LOG(2):GOSUB 290
150 PRINT:PRINT " Probabilite d'insucces
 a l'essai no.";i;":";:z=pins:GOSUB 290
160 PRINT " Gain d'information associe:"
;:z=LOG(n1/n2)/LOG(2):GOSUB 290
170 g=ps*LOG(n1)/LOG(2)+pins*LOG(n1/n2)/
LOG(2)
180 PRINT:PRINT " Gain espere d'informat
ion de l'essai no.";i;"apres";i-1;"echec
(s):";:z=g:GOSUB 290
190 rg=(n+1)/2
200 PRINT " Rang moyen de l'essai qui re
ussit:";rg
210 PRINT " Gain moyen d'information par
 essai:";:z=LOG(n)/LOG(2)/rg:GOSUB 290
220 PRINT:PRINT "Voulez vous continuer (
o ou n) ?";
230 r$=""
240 WHILE r$="" OR (LOWER$(r$)<>"o" AND
LOWER$(r$)<>"n")
250 r$=INKEY$
```

```
260 WEND
270 PRINT r$
280 IF LOWER$(r$)="o" THEN CLEAR:GOTO 30
 ELSE END
290 '
300 '          ARRONDI A D DECIMALES
310 '
320 z=ROUND(z,d):PRINT z
330 RETURN
```

Décomposition des mots en lettres

Le joueur B peut utiliser une information importante relative à la structure des mots : ils sont formés des 26 lettres de l'alphabet. Une deuxième méthode consistera donc à poser des questions du type :

la première lettre du mot choisi est-elle un C ?

Nous pouvons reprendre, à cette occasion, l'analyse du paragraphe précédent. Cette fois, pour la première lettre, le domaine des choix possibles s'exprime par :

N=26 soit 4,7 bits

La quantité d'informations attendue de la première question vaut, selon notre programme SHANNON :

G=0,235 bits

Le nombre de bits par question augmentera avec le nombre de ces questions, pour les raisons déjà évoquées. La valeur moyenne est voisine de 0,35 bit. Trouver la première lettre exigera donc, en moyenne, 13,5 questions.

Un mot français contient en moyenne six lettres, auxquelles il faut ajouter un symbole définissant la fin du mot. Au total, 7*13,5=94,5 questions seront nécessaires, en ordre de grandeur, pour trouver le mot inconnu.

La deuxième méthode apparaît ainsi 170 fois plus efficace que la première. Pourquoi ce progrès considérable ? *Nous avons utilisé une information complémentaire relative à la structure du problème :* **les mots sont formés de lettres.** Le système étudié est donc **séparable.** Les variables essentielles et les objectifs peuvent être "éclatés". Le gain unitaire d'information obtenu à chaque étape de la recherche, c'est-à-dire à chaque question, est ainsi substantiellement amélioré.

Dichotomie utilisant l'ordre alphabétique

Il existe une **liste ordonnée** des mots français : le dictionnaire. La mise en œuvre de cette connaissance supplémentaire peut, nous allons

le voir, accroître encore l'efficacité de la recherche. La première question sera :

> le mot choisi est-il dans la première moitié du dictionnaire ?

(par exemple dans les 500 premières pages si le volume en comporte 1 000).

La réponse (oui ou non) écarte, dans les deux cas, la moitié des mots. Le gain d'information est donc de 1 bit. Une deuxième, puis une troisième question du même type seront alors posées. En définitive, en 15 questions exactement, valant chacune 1 bit, le mot inconnu sera trouvé. La quantité d'information acquise, soit 15 bits, correspond bien au niveau d'incertitude du problème.

Ici nous atteignons un optimum absolu. Une question à réponse binaire ne peut apporter plus d'un bit d'information. 15 questions de ce type seront donc nécessaires, au minimum, pour toute méthode, si l'on veut résoudre un problème valant 15 bits. L'optimum, nous l'avons dit, a été obtenu grâce à la connaissance et à l'emploi d'une structure d'ordre total, constituée par la liste des mots rangés par ordre alphabétique dans le dictionnaire.

Dichotomie utilisant l'ordre des lettres

Nous pouvons combiner la dichotomie et la deuxième méthode. Le mot sera alors défini, lettre par lettre, grâce à des questions du type :

> la lettre initiale fait-elle partie des 13 premières lettres de l'alphabet latin ?

Cinq questions de ce type identifieront une lettre. 35 questions, en moyenne, suffiront pour la découverte d'un mot. Ici, les sources d'efficacité sont doubles :

- décomposition du mot en lettres (éclatement des objectifs) ;
- liste **ordonnée** des lettres représentée par l'alphabet (structure d'ordre).

La quatrième méthode apparaît cependant moins efficace que la troisième (35 questions au lieu de 15). Le domaine de recherche associé aux arrangements possibles des lettres (2^35 environ) contient, en effet, beaucoup plus d'éléments que le dictionnaire des mots français (2^15 environ). Par exemple, le mot XHCT n'existe pas. Au premier domaine correspond une quantité d'incertitude de 35 bits contre 15 seulement au second. La décomposition en lettres, efficace au niveau du gain d'information par question posée, présente l'inconvénient d'accroître la quantité totale d'information nécessaire à la solution du problème.

Retour au cambrioleur de coffres-forts

Revenons un instant à une situation évoquée au début de ce chapitre. Par commodité (information exprimée en nombres entiers de bits), supposons un cambrioleur placé, sans moyen particulier, devant un coffre-fort équipé de 4 boutons à 8 chiffres chacun. Le nombre des combinaisons possible est donc 8^4=2^12=4096. La quantité d'information associée à l'ouverture du coffre est donc égale à 12 bits. Par analogie avec le problème du dictionnaire, nous pouvons classer, comme suit, les méthodes utilisables.

1re méthode

Afficher au hasard un chiffre sur chaque bouton, puis essayer d'ouvrir le coffre. L'opération est un essai, une recherche d'information identique à une question. La réponse est binaire :

oui, si le coffre s'ouvre ;
non, dans le cas contraire.

Dans le second cas, la combinaison, essayée en vain, est éliminée ; un autre essai est tenté et ainsi de suite. Les gains d'information espérés à chaque essai peuvent être calculés par le programme SHANNON. La première tentative vaut, sensiblement, 0,0032 bits. Les autres tentatives valent un peu plus. La valeur moyenne avoisine 0,0059 bits et il faudra environ 2 050 essais.

2e méthode

Cette fois, le cambrioleur muni, par exemple, d'un stéthoscope, peut repérer, pour chaque bouton, le chiffre qui appartient à la combinaison recherchée. Les questions équivalentes sont du type :

le premier chiffre est-il un trois ?

Le gain moyen d'information associé à une première question de cette sorte est 0,544 bit. Ce gain unitaire (par essai) croît avec le nombre des essais entre 0,544 et 1 bit. Sa valeur moyenne est 0,667 bits et le nombre moyen d'essais nécessaires sera égal à 4,5 bits. L'ouverture du coffre, associée à une quantité d'incertitude de 12 bits, sera donc effectuée en 4,5*4=18 essais approximativement.

Le rapport d'efficacité, 114 environ, entre les gains d'information par essai, obtenus dans la deuxième et la première méthodes, est évidemment dû à la reconnaissance d'une structure **séparable** du problème initial (éclatement des objectifs) et à la mise en œuvre d'une technique de recherche d'information adaptée à cette structure.

Il serait possible d'aller beaucoup plus loin dans l'exposé des applications de la théorie de l'information à la résolution de problèmes, mais tel n'est pas le sujet de cet ouvrage. Notre but était seulement d'éveiller l'intérêt du lecteur pour ce domaine.

DEUXIÈME PARTIE
JEUX DE RÉFLEXION

Chapitre 5
JEUX DANS LES ARBRES

JEUX ET INTELLIGENCE ARTIFICIELLE

Dès que la possibilité de simuler certaines opérations intellectuelles cessa d'être un rêve fou, les jeux de stratégie (échecs, dames, etc.) exercèrent sur les chercheurs une réelle fascination.

Ils jouissent, en effet, d'un "statut intellectuel" élevé. Les championnats d'échecs, on le sait, sont, dans certains pays, affaires d'Etat : prestige et réputation sont en jeu. Ce sont, encore, des problèmes bien posés : le terrain de jeu, les pièces, les mouvements sont parfaitement définis et connus de tous, parfois depuis des siècles. Les règles sont précises et les résultats indiscutables : une partie est gagnée, nulle ou perdue. Le hasard n'y joue aucun rôle. Enfin, leur structure est arborescente : chaque coup joué, choisi parmi un nombre limité de possibles, crée une nouvelle situation et borne l'éventail des mouvements ultérieurs. Les jeux de stratégie offrent donc d'excellents exemples de structures hiérarchisées. Les méthodes mises au point à leur propos sont généralisables.

Les premiers travaux apparaissent dans l'immédiat après-guerre. En 1945, K. Zuse, l'un des pionniers de l'informatique naissante, programme les règles du jeu d'échecs. Le 9 mars 1949, le mathématicien américain Claude Shannon, père de la théorie de l'information, tient conférence à New York sous le titre "Programmer un ordinateur pour jouer aux échecs". Fait remarquable : nombre d'idées originelles de Shannon se retrouvent dans les programmes actuels. L'année suivante, le grand logicien anglais Alan Turing propose, à son tour, un programme d'échecs simulé à la main.

Ces tentatives sont suivies d'essais programmés sur les premiers ordinateurs : Kister et al., sur MANIAC-I en 1956, Bernstein et al., sur IBM 704 en 1957, Newell, Shaw et Simon sur RAND JOHNIAC en 1958. Mais le premier programme vraiment performant, MACHACK VI, est écrit au MIT, en 1967, par Richard Greenblatt, encore étudiant, et deux de ses collègues.

Un programme très célèbre concernant les dames (la variété anglo-saxonne porte le nom de checkers) est écrit par Arthur Samuel en 1952. Une seconde version, en 1954, est suivie, l'année d'après, d'une troisième, incorporant le premier mécanisme d'apprentissage. Le chapitre 7 de cet ouvrage lui est consacré.

Créer des programmes de jeu capables de vaincre les champions humains constitua, dès l'origine, un formidable défi, Nous avons quelque peine aujourd'hui à le comprendre, blasés que nous sommes par l'abondance des programmes commercialisés. Reportons-nous, pourtant, par la pensée, trente ans en arrière et imaginons l'invraisemblable, l'impensable : jouer aux échecs, aux dames, au backgammon, etc., *avec un ordinateur et se faire battre !* Dans un chapitre de l'ouvrage *Computers and thought,* paru en 1963, Newell, Shaw et Simon écrivaient, à propos des échecs : "Le jeu est suffisamment profond et subtil dans ses implications pour avoir suscité une classe de professionnels et permis une analyse approfondie, menée à travers deux cents ans d'études et de jeux intensifs sans épuiser sa richesse. De telles caractéristiques le désignent comme arène naturelle pour des essais de mécanisation. Construire une machine qui joue avec succès aux échecs signifierait, semble-t-il, pénétrer au cœur des activités intellectuelles humaines".

Le même Simon (prix Nobel d'Economie en 1978), avait pleine confiance dans la réussite de l'opération : il pariait, en 1958, qu'avant dix ans, un ordinateur dominerait le championnat du monde d'échecs. Ce pari a été perdu, mais les meilleurs programmes d'échecs battent aujourd'hui très largement les joueurs moyens. Ils se situent à 2 200 points ELO (échelle servant au classement international des joueurs) contre 2 400 environ pour les très Grands Maîtres. Selon certains, le champion du monde d'échecs sera, avant la fin du siècle, un ordinateur (ou plutôt un programme).

Le programme de dames de Samuel a vaincu de grands joueurs, mais les plus incontestables succès furent obtenus pour le backgammon (programme BKG 9.8 de Hans Berliner à l'Université de Carnegie Mellon, aux Etats-Unis) et Othello (programme The Moor écrit par Mike Reeve au College Imperial de l'Université de Londres). Ces programmes ont effectivement vaincu des champions du monde en titre. BKG 9.8 gagna, par 7 points à 1, le 15 juillet 1979, à Monte-Carlo, contre le champion du monde de backgammon : Luigi Villa. The Moor gagna, au Japon, le 19 juin 1980, par 36 à 27, contre le champion du monde d'Othello : Hiroshi Inoue.

Il n'est donc pas sans intérêt d'analyser, dans leurs principes et leurs premiers pas, les méthodes qui ont permis ces victoires (des chercheurs et programmeurs). Tel est l'objet principal du présent chapitre. L'application en vraie grandeur sera faite à l'awele, au chapitre suivant.

JEUX DE RÉFLEXION A UNE PERSONNE

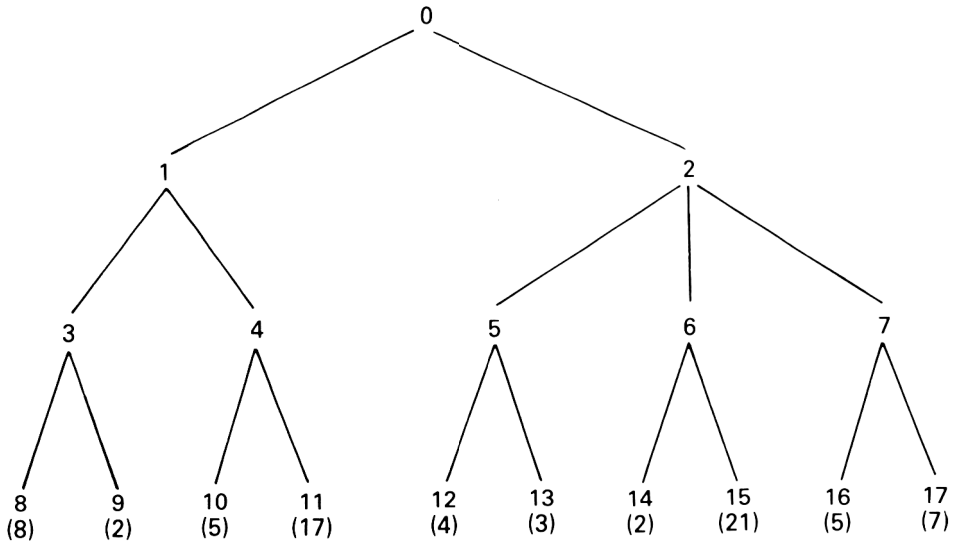
Le chapitre 1 a présenté, en avant-première, un exemple de tels jeux : le TAQUIN, avec trois programmes associés à trois méthodes différentes de recherche. Nous rencontrerons ici des notions plus générales :

- positions, mouvements et leurs représentations informatiques ;
- graphes ou arbres de jeu ;
- fonctions d'évaluation ;
- méthodes d'exploration du graphe et de recherche des chemins optimaux.

Dans le même temps, nous bâtirons deux programmes simples : JEU1 ET JEU2. Ils nous permettront d'expérimenter sur notre Amstrad et de rendre ces idées plus concrètes.

Représentations d'un jeu

Un jeu peut être décrit par les positions possibles, les règles qui définissent les mouvements légaux et l'objectif poursuivi. Nous devrons par ailleurs, et ce point est essentiel, traduire ces notions dans un langage de communication avec l'ordinateur. Une représentation visuelle est commode. En voici un exemple :



Il symbolise un graphe formé de sommets et d'arcs. Chaque sommet est associé à une position. Le sommet 0 (la racine) est la position actuelle à partir de laquelle on doit jouer. Les arcs issus d'un sommet I

figurent les coups permis à partir de la position associée à ce sommet. Ils aboutissent à des sommets dits successeurs (ou fils ou descendants) du sommet I, désigné lui-même comme leur père ou ascendant. Un graphe tel que tout sommet possède un unique ascendant est un arbre. Les sommets terminaux sont des "feuilles". Certaines sont "plus égales" que d'autres. Elles représentent le ou les points d'arrivée préférentiels, le ou les buts du jeu. Pour notre arbre exemple, aux sommets terminaux sont attachées des valeurs (chiffres entre parenthèses).

Le traitement du jeu de TAQUIN utilisa (cf. chapitre 1) des chaînes de caractères contenant 9 chiffres (1 à 8 plus un 9 pour la case vide) afin de représenter les positions. Un générateur de coups, le sous-programme SUCCESSEURS, calculait, pour toute configuration donnée, le nombre des successeurs, les numéros d'ordre des positions filles et le contenu de celles-ci. Dans le cas de notre exemple graphique, il conviendra de numéroter les sommets, par niveaux successifs, de 0 à 17. Comment symboliser ensuite, dans une structure informatique, arcs et successeurs ? Une réponse possible, en BASIC, utilisera des tables. Si I est le numéro repère d'un sommet et J le rang (numéro d'ordre de 1 à...) d'un successeur,

- la table NS(I) contiendra le nombre de successeurs du sommet I ;
- la table SU(I,J) mémorisera les repères des successeurs de rang J du sommet I.

Ces tables nous permettront de cheminer dans le graphe, à partir d'un sommet quelconque, vers ses successeurs directs... et les descendants de ceux-ci, de proche en proche.

Pour achever la définition du jeu très schématique, figuré par notre graphe, nous devons préciser l'objectif. Il s'agira d'atteindre une ou plusieurs positions terminales (feuilles). Nous leur avons affecté des valeurs numériques entières (chiffres entre parenthèses sur le graphe) mémorisées dans une table VA(I). Donnons-nous comme but la recherche d'une position terminale de valeur maximale. Le programme JEU1 devra résoudre ce problème. Les instructions 50 à 160, listées ci-dessous, initialisent les tables NS(I), SU(I,J), VA(I) et numérotent les sommets (variable N) à partir de la racine (repère RA=0). Les données sont définies en DATA et modifiables à volonté.

```
50 DEFINT a-z
60 DIM ns(7),su(7,3),va(17)
70 ra=0
80 DATA 2,2,3,2,2,2,2,2
90 FOR i=0 TO 7:READ ns(i):NEXT
100 FOR i=0 TO 7
110    FOR j=1 TO ns(i)
120       n=n+1:su(i,j)=n
```

```
130    NEXT j
140 NEXT i
150 DATA 8,2,5,17,4,3,2,21,5,7
160 FOR i=8 TO 17:READ va(i):NEXT
```

Recherche du maximum

L'arbre de jeu pris comme exemple est analysable en une suite de niveaux. Au niveau 0 est la racine dont les successeurs forment le niveau 1. Le niveau 2 rassemble les successeurs de ceux-ci... et ainsi de suite jusqu'à l'horizon (niveau H). Nous cherchons un **algorithme,** nom donné à toute méthode systématique résolvant le problème à coup sûr. Le mécanisme général en est clair. Les valorisations significatives affectent les positions terminales, de niveau 3 dans notre exemple. Nous devons donc partir de celles-ci et remonter vers la racine du graphe (niveau 0). Dans cet esprit, nous **marquerons** chaque sommet de niveau 2 d'un nombre égal au maximum des valeurs prises par les successeurs de ce sommet. Nous obtenons le tableau suivant :

Sommet	*Valeur*
3	MAX(8,2)=8
4	MAX(5,17)=17
5	MAX(4,3)=4
6	MAX(2,21)=21
7	MAX(5,7)=7

Itérons l'opération pour les sommets de niveau 1. Cela donne :

Sommet	*Valeur*
1	MAX(8,17)=17
2	MAX(4,21,7)=21

Enfin, la racine (sommet 0) sera marquée du maximum des valeurs marquées pour ses deux successeurs, soit :

MAX(17,21)=21

Le maximum, notons-le, correspond au successeur 2. Le coup à jouer, à partir de la position initiale 0, est donc l'arc (0,2). De même, la descente de l'arbre définira les chemins (2,6) et (6,15) vers l'optimum. Les résultats sont donc :

Chemin optimal : 0,2,6,15
Valeur optimale : 21

Reste maintenant une tâche un peu délicate : traduire en programme ce cheminement intellectuel. Il serait en effet impraticable, pour des graphes de dimensions réalistes, de mémoriser toutes les positions et leurs valeurs. Il convient de définir une méthode efficace, qui économise le temps de calcul et la mémoire.

Nous tenterons de l'obtenir en choisissant :

- une règle d'exploration de l'arbre, sommet par sommet, sans oubli ni répétition ;
- une mémorisation, à chaque niveau, du seul sommet dont nous aurons besoin, à un instant donné, pour générer les successeurs, du nombre de ces successeurs et de la valeur temporaire du maximum.

Pour ce faire, nous pouvons associer, à chaque niveau du graphe, à partir de la racine, une boucle FOR...NEXT qui balaiera les successeurs des sommets de ce niveau. Ces boucles seront imbriquées et en nombre égal au nombre de sommets non terminaux (soit 3 dans notre exemple). Présentons d'abord la liste des variables et le programme JEU1, réservant les commentaires pour la suite.

Liste des variables

Tables :

NS(7)	: nombre des successeurs d'un sommet.
SU(7,3)	: numéros repères des successeurs par sommet et rang.
VA(17)	: valorisations des sommets.

Variables :

EV0, EV1, EV2, EV3	: maxima à chaque niveau.
I, J, J1, J2, J3	: indices des boucles.
N	: numéros repères des sommets.
NS1, NS2, NS3	: nombres courants de successeurs à chaque niveau.
RA	: numéro repère de la racine.
S0, S1, S2, S3	: repères des sommets courants à chaque niveau.
SOPT1, SOPT2, SOPT3	: repères des sommets qui balisent le chemin optimal.

Listing du programme JEU1

```
10 MODE 1:INK 0,13:INK 1,0:PAPER 0:PEN 1
:CLS:PRINT:PRINT TAB(18)"JEU1":PRINT
20 PRINT " Ce programme calcule le chemi
n optimal"
30 PRINT "sur un graphe a 3 niveaux"
40 PRINT "defini par des DATA.":PRINT
50 DEFINT a-z
60 DIM ns(7),su(7,3),va(17)
```

```
70 ra=0
80 DATA 2,2,3,2,2,2,2,2
90 FOR i=0 TO 7:READ ns(i):NEXT
100 FOR i=0 TO 7
110   FOR j=1 TO ns(i)
120     n=n+1:su(i,j)=n
130   NEXT j
140 NEXT i
150 DATA 8,2,5,17,4,3,2,21,5,7
160 FOR i=8 TO 17:READ va(i):NEXT
170 GOSUB 240:'Maximum
180 PRINT " Chemin optimal:";ra;sopt1;so
pt2;sopt3
190 PRINT " Valeur optimale:";ev0
200 END
210 '
220 '        MAXIMUM
230 '
240 ev0=-30000:ev1=-30000:ev2=-30000
250 s0=ra:ns1=ns(ra)
260 FOR j1=1 TO ns1
270   s1=su(s0,j1):ns2=ns(s1)
280   FOR j2=1 TO ns2
290     s2=su(s1,j2):ns3=ns(s2)
300     FOR j3=1 TO ns3
310       s3=su(s2,j3)
320       ev3=va(s3)
330       IF ev3>ev2 THEN ev2=ev3:sopt3=
s3
340     NEXT j3
350     IF ev2>ev1 THEN ev1=ev2:sopt2=s2
360   NEXT j2
370   IF ev1>ev0 THEN ev0=ev1:sopt1=s1
380 NEXT j1
390 RETURN
```

Suivre le déroulement des opérations dans le sous-programme MAXIMUM n'est pas sans intérêt. L'arbre de jeu est d'abord parcouru selon le chemin (0, 1, 3) dont les sommets sont mémorisés en S0, S1, S2. A ce point, la boucle la plus interne balaie les successeurs du sommet 3, note les valeurs y associées, calcule le maximum EV2=8 pour les sommets 8 et 9 et mémorise provisoirement en SOPT3=8 le repère du sommet d'évaluation maximale. La valeur initiale EV2=−30000 est prise inférieure à toutes les valorisations possibles des sommets du graphe afin d'assurer un calcul correct du maximum.

A la sortie de la troisième boucle, le maximum provisoire obtenu est "remonté" en EV1=8 au niveau 1. Le numéro SOPT2=3 du sommet de niveau 2 situé à cet instant sur un chemin optimal est mémorisé.

Puis (2e tour de la 2e boucle), le programme calcule le numéro repère du 2e successeur, soit 4, du sommet S1 (ici le sommet 1) qu'il avait conservé en mémoire. La 3e boucle tourne à nouveau. Elle détermine le maximum, soit EV2=17, des sommets 10 et 11 et la nouvelle valeur SOPT3=11. A la sortie, EV1 (ici 8) est comparé à la nouvelle valeur de EV2, soit 17. Le maximum 17 est remonté en EV1=17. Ensuite, avant de quitter le premier successeur du sommet 0, le maximum provisoire EV1=17 est copié sur EV0 avec JSOPT1=1. Le reste du mode opératoire se construit aisément.

Vous pouvez maintenant entrer et lancer le programme JEU1. Le résultat devrait ressembler à la copie d'écran ci-dessous.

```
                    JEU1

 Ce programme calcule le chemin optimal
sur un graphe a 3 niveaux
defini par des DATA.

 Chemin optimal: 0  2  6  15
 Valeur optimale: 21
```

Rien n'interdit d'expérimenter sur d'autres valeurs et des graphes différents afin d'explorer les mécanismes de l'algorithme.

Lors de la mise en œuvre d'un jeu réel, les instructions du type :

```
280     S1=SU(S0,J1) : NS2=NS(S1)
```

sont évidemment remplacées par l'appel à un sous-programme générateur qui détermine les coups possibles à partir d'une position définie et leur nombre. Reportez-vous, s'il vous plaît, à titre d'exemple, au programme TAQUIN3 du chapitre 1.

Une variante

L'exploration des graphes et arbres est une technique importante de l'Intelligence artificielle. A ce titre, elle doit être bien comprise. Nombre de problèmes présentent une structure arborescente : chaque élément d'information renvoie à d'autres (pères, fils ou frères).

L'usage de boucles imbriquées est commode. Elles offrent, en BASIC, un cadre clair, tout préparé et des temps de calcul relativement courts. Nous nous en servirons abondamment au cours de cette partie de l'ouvrage. Mais les programmes qui les utilisent, tels JEU1, manquent de généralité. Certaines instructions se répètent dans chaque boucle. Il

serait agréable de paramétrer l'horizon sans devoir modifier le programme. Pour ces raisons, nous avons construit le programme JEU2. Il résout les mêmes problèmes que JEU1, pour un horizon H variable et sans boucles explicites.

Les mouvements à l'intérieur d'un arbre dépendent de deux paramètres : le niveau P dans l'arborescence et le rang J(P) du successeur, à ce niveau P du sommet "père" S(P−1). Par ailleurs, trois types de mouvements et trois seulement sont possibles :

a) Descente de l'arbre : la main passe au niveau P+1 et au premier successeur (de rang 1) du sommet S(P).

b) Exploration horizontale au même niveau P : le programme saute au "frère", de rang supérieur, du sommet S(P) et donc J(P) est incrémenté de 1.

c) Remontée à l'ascendant direct (le "père").

Il existe plusieurs types d'explorations qui combinent ces mouvements de diverses manières. Le mode qui nous conviendrait doit parvenir très rapidement au dernier niveau (l'horizon) qui contient les valorisations. On parle alors de "postordre", par opposition au préordre qui chemine à partir de la racine. Les conditions génératrices de chaque sorte de mouvement sont donc :

a) **Descente** : le dernier niveau n'est pas atteint (P<H) et le niveau immédiatement inférieur n'a pas été complètement exploré.

b) **Déplacement horizontal** : le dernier niveau est atteint ou le niveau inférieur a été complètement exploré et il reste des sommets à visiter au niveau actuel.

c) **Remontée** : aucune des conditions précédentes n'est vérifiée.

D'autre part, au dernier niveau (horizon), il convient de noter la valeur de chaque sommet. Si elle est supérieure à la dernière évaluation connue au niveau H−1, ce maximum provisoire est "remonté" aussi haut que possible. Les sommets situés sur le chemin optimal sont mémorisés.

Ces commentaires vous permettront, je l'espère, de bien comprendre le mécanisme du programme JEU2 (grand dévoreur de tables), dont le listing suit, précédé de la liste des variables.

Liste des variables

Tables :

EV(H)	: dernier maximum à chaque niveau P.
J(H+1)	: rang du sommet courant au niveau P.
NS(NMAX)	: nombre réel de successeurs de chaque sommet de l'arbre.

NSUC(H+1)	: nombre de successeurs du sommet courant au niveau P.
S(H)	: sommet courant au niveau P.
SOPT(H)	: sommet sur le chemin optimal au niveau P.
SU(NMAX, NSMAX)	: numéros repères des successeurs de chaque sommet de l'arbre.
VA(NMAX)	: valorisations des sommets terminaux.

Variables simples :

H	: horizon en nombre de niveaux.
I, J	: indices de boucles.
N	: numéro repère d'un sommet de l'arbre.
NMAX	: nombre maximal de sommets de l'arbre.
NSMAX	: nombre maximal de successeurs possibles pour un sommet de l'arbre.
P	: niveau courant.
R	: variable de stockage du niveau P.
RA	: repère de la racine de l'arbre.

Listing du programme JEU2

```
10 MODE 1:INK 0,13:INK 1,0:PAPER 0:PEN 1
:CLS:PRINT:PRINT TAB(18)"JEU2":PRINT
20 PRINT " Ce programme calcule le chemi
n optimal"
30 PRINT "sur un graphe a 3 niveaux"
40 PRINT "defini par des DATA.":PRINT
50 DEFINT a-z:ra=0:h=3:nmax=17:nsmax=3
60 DIM ns(nmax),su(nmax,nsmax),va(nmax),
ev(h),j(h+1),nsuc(h+1),s(h),sopt(h)
70 DATA 2,2,3,2,2,2,2,2
80 FOR i=0 TO 7:READ ns(i):NEXT
90 FOR i=0 TO 7
100   FOR j=1 TO ns(i)
110     n=n+1:su(i,j)=n
120   NEXT j
130 NEXT i
140 DATA 8,2,5,17,4,3,2,21,5,7
150 FOR i=8 TO nmax:READ va(i):NEXT
160 GOSUB 240:'Maximum
170 PRINT " Chemin optimal:";ra;:FOR i=1
 TO h:PRINT sopt(i);:NEXT:PRINT
180 PRINT " Valeur optimale:";ev(0):PRIN
T
190 END
```

```
200 '
210 '          MAXIMUM
220 '
230 ' ------ Initialisation
240 FOR i=0 TO h-1:ev(i)=-30000:NEXT
250 s(0)=ra:p=1:nsuc(p)=ns(ra):j(p)=0
260 ' ------ Boucle de calcul
270 j(p)=j(p)+1:s(p)=su(s(p-1),j(p)):nsu
c(p+1)=ns(s(p))
280 IF p=h THEN GOSUB 370:'Chemin optima
l
290 IF p<h AND j(p+1)<nsuc(p+1) THEN p=p
+1:GOTO 270
300 IF j(p)<nsuc(p) THEN 270
310 j(p)=0:p=p-1:IF p>0 AND j(p)>=nsuc(p
) THEN 310
320 IF p>0 THEN 270
330 RETURN
340 '
350 '          CHEMIN OPTIMAL
360 '
370 ev(p)=va(s(p)):r=p
380 IF ev(r)>ev(r-1) THEN ev(r-1)=ev(r):
sopt(r)=s(r)
390 r=r-1:IF r>0 THEN 380
400 RETURN
```

Horizon et fonction d'évaluation

En théorie, les programmes du type JEU1 ou JEU2 permettent de résoudre n'importe quel puzzle. Il "suffit" de dérouler le nombre nécessaire de boucles pour atteindre le niveau terminal où se reconnaissent succès et échecs. L'algorithme trace alors le (ou les) chemin(s) optimal(aux) à partir de la position actuelle.

En pratique, se rencontre très vite le mur de l'explosion combinatoire. Prenons l'exemple, déjà bien connu, du TAQUIN. A chaque stade, le nombre moyen de coups possibles avoisine 3. C'est également le nombre moyen de successeurs d'un sommet de l'arbre représentatif du jeu. Une exploration complète de cet arbre sur un horizon modeste, cinquante niveaux (et autant de boucles), conduirait à 3^50=7,18*10^23 positions terminales. Admettons, avec quelque optimisme, la cadence de mille boucles par seconde pour notre Amstrad. Une solution du problème, si elle existe, sera trouvée en 7,18*10^20 secondes soit 22 767 milliards d'années. Combien mesquin, sous cet éclairage, apparaît notre pauvre univers avec ses 15 milliards d'années d'existence ?

Il faut, apparemment, trouver autre chose. Selon la terminologie du chapitre 4, les jeux considérés forment des systèmes arborescents et

intégrés. Une stratégie, en fait la seule possible, est de les transformer en systèmes **presque hiérarchisés** dont la solution est construite par pièces et morceaux.

Cela implique le choix :

- d'un horizon de calcul ;
- d'objectifs "éclatés" ; la forme la plus courante est une fonction d'évaluation qui permet de valoriser chaque position terminale à l'horizon défini.

L'un ou l'autre choix ne peuvent être rigoureux. Il s'agit d'**heuristiques** : méthodes systématiques mais dont le résultat n'est pas garanti. La fonction d'évaluation doit naturellement garder un rapport étroit avec le but du jeu. Sa construction vise à condenser l'expertise disponible, l'expérience des meilleurs. Le succès en ce domaine conditionne évidemment la qualité du programme.

Une fois définis l'horizon et la fonction d'évaluation, l'algorithme d'optimisation est appelé de façon répétitive. Le meilleur coup est calculé à partir de la position actuelle et joué. A l'horizon de la position ainsi créée, apparaissent d'autre configurations terminales. Elles sont, à leur tour, évaluées et ainsi de suite.

L'analyse du TAQUIN a mis en œuvre plusieurs exemples de fonctions d'évaluation (distance Manhattan et ses dérivés). Nous en rencontrerons d'autres dans cette troisième partie, à propos de l'awele et du jeu de dames.

JEUX DE RÉFLEXION A DEUX PERSONNES

Les règles d'un jeu de ce type sont précises et ne laissent au hasard aucune place. Elles définissent complètement les mouvements permis en toute occurrence. Chaque joueur devra, à tour de rôle, réaliser un coup choisi parmi ceux autorisés, et la suite des choix successifs constituera une partie. Celle-ci est terminée par l'une des situations suivantes (configurations terminales), définies sans ambiguïté :

- gain du joueur A et perte pour le joueur B ;
- gain du joueur B et perte pour le joueur A ;
- partie nulle.

Le problème posé peut être ainsi caractérisé. C'est un duel (deux joueurs) défini par une arborescence (succession des choix) à information parfaite (à chaque instant, la situation est totalement connue). Les positions intermédiaires ne peuvent être valorisées par les seules règles du jeu, la valeur des configurations terminales (gain, perte ou partie nulle), étant seule connue. Les intérêts des deux joueurs sont strictement opposés : le jeu est dit à somme nulle.

L'arbre des choix

La désignation du joueur (A par exemple) qui exécutera le premier mouvement, fixe la position initiale. A partir d'elle, ce joueur dispose de plusieurs choix d'une liste parfaitement précise. Chacun d'eux définit une nouvelle configuration qu'il appartient au joueur B de modifier, selon diverses directions en nombre limité. De proche en proche, l'ensemble des configurations possibles (en cours et en fin de partie) peut être décrit par un arbre de jeu, partant d'un sommet initial et se développant sur plusieurs niveaux (autant de niveaux que de coups). Chaque sommet de l'arbre définit une position.

Nous pouvons programmer un ordinateur afin qu'à partir des règles du jeu, il détermine toutes les configurations qui succèdent à une configuration donnée ou, inversement, celles qui la précèdent. Ces positions peuvent, en principe, être stockées.

Faisons un rêve. Imaginons que la capacité de mémoire et la vitesse de calcul de notre ordinateur soient telles qu'il puisse, à partir de la position initiale, déterminer de proche en proche et mémoriser toutes les positions possibles jusqu'aux configurations terminales. Ces dernières peuvent être toutes valorisées. Elles seront, par exemple, marquées des nombres $+1, 0, -1$, selon que la partie, terminée sur cette configuration, est, pour le joueur 1, gagnée, nulle ou perdue.

Il est maintenant possible de marquer les positions qui précèdent, immédiatement, les configurations terminales et de les désigner, par conséquent, comme gagnantes, perdantes ou nulles. Voyons cela avec quelque détail.

Soit $C(n-1)$ une configuration, qui, après un coup du joueur A, et selon le mouvement choisi, se transforme en l'une des configurations :

$C(n)$, $C'(n)$, $C''(n)$ **toutes terminales.**

Si l'une au moins des $C(n)$, par exemple $C'(n)$ est gagnante pour le joueur A (marquée de la valeur $+1$), $C(n-1)$ sera aussi gagnante. Il suffit, en effet, qu'au stade $C(n-1)$, le joueur A joue pour passer de $C(n-1)$ à $C'(n)$. De même, si l'une au moins des $C(n)$ est marquée d'un zéro (partie nulle), toutes les autres étant perdantes, $C(n-1)$ sera marquée d'un zéro. Enfin, si toutes les $C(n)$ sont perdantes, $C(n-1)$ sera perdante (marquée de -1).

Examinons maintenant le cas d'un mouvement du joueur B qui transforme $C(n-1)$ en l'une des configurations $C(n)$, toutes terminales et donc marquées. Si l'une au moins des $C(n)$, soit $C''(n)$, est marquée de -1 (perdante pour A et donc gagnante pour B), $C(n-1)$ sera marquée de -1 car le joueur B, placé en $C(n-1)$, pourra toujours gagner en allant de $C(n-1)$ à $C''(n)$. Si l'une au moins des $C(n)$ est marquée de 0, les autres étant marquées par des nombres 0 ou $+1$, $C(n-1)$ sera marquée de 0 (partie nulle). Enfin, si toutes les $C(n)$ sont marquées de $+1$, $C(n-1)$ sera marquée de $+1$ (gain de A et donc perte de B).

Nous pouvons condenser ainsi les règles de valorisation ou marquage de C(n−1) selon qu'il appartient à A ou B de jouer, à partir de cette configuration :

a) **A doit jouer**

C(n−1) est marquée de la valeur **maximale** associée aux valorisations des configurations terminales C(n) qui lui succèdent. De plus, on retiendra (mise en mémoire) le ou les mouvements de A qui assurent ce maximum.

b) **B doit jouer**

C(n−1) est marquée, dans les mêmes conditions, de la valeur **minimale** associée aux valorisations des configurations terminales qui lui succèdent. Le ou les mouvements de B qui assurent ce minimum sont mémorisés.

Après avoir ainsi marqué les configurations "pré-terminales" (C(n−1), le processus peut, à l'évidence, être répété. Il marque ainsi toutes les configurations C(n−2) qui précèdent les configurations C(n) et C(n−1), déjà valorisées. De proche en proche, toutes les configurations peuvent être marquées, jusqu'à la configuration initiale inclusivement. Cette dernière valorisation peut dépendre du trait (mouvement initial), selon qu'il appartient au joueur A ou au joueur B. De plus, chacun des mouvements optimaux ayant été enregistré, la suite des coups successifs qui assurent gain, perte ou partie nulle (selon la valeur de la position initiale) est complètement définie. En conséquence, la seule indication du joueur qui a le trait permettrait, au terme de ces opérations, d'afficher le résultat en toute certitude.

Le théorème de Lucas

Les considérations précédentes conduisent à une proposition aperçue par Lucas en 1896 et démontrée par le mathématicien Zermelo en 1912. Elles s'appliquent à tous les jeux de réflexion pure et, en particulier, aux dames, échecs, othello, go, etc. Il est donc assuré qu'à tout instant d'un tel jeu, et, en particulier au départ, il existe un "équilibre" (résultat final) tel qu'aucun des deux joueurs ne puisse obtenir un résultat plus favorable contre un adversaire parfaitement clairvoyant. Avant que les joueurs ne s'assayent à une table d'échecs, cet équilibre existe et correspond à l'une seulement des trois situations suivantes : les Blancs gagnent, la partie est nulle, les Noirs gagnent.

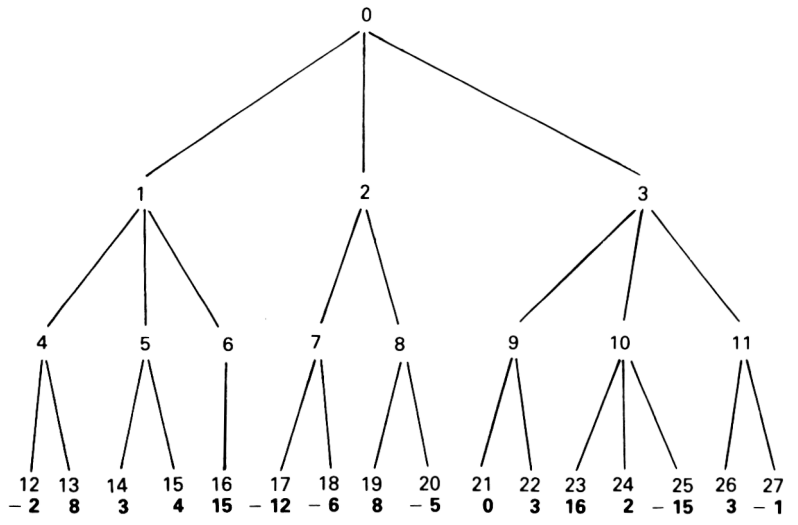
Pourquoi, alors, **la** solution des échecs n'est-elle pas connue ? La réponse tient encore à l'explosion combinatoire. Lors de sa conférence de 1949, rappelée plus haut, Shannon calculait qu'il y a environ 10^120 parties d'échecs possibles de 40 coups (durée moyenne d'une partie de maîtres). Or ce nombre dépasse tellement celui des atomes de l'univers que, même si chaque atome était remplacé par le plus puissant ordinateur connu (le Cray 1 par exemple), le système tout entier mettrait un temps considérable à jouer son premier coup.

La conclusion qui se dégage est, à première vue, paradoxale. Une vérité existe que, selon toute vraisemblance, nous ne connaîtrons jamais. La dimension combinatoire des jeux usuels les met à l'abri d'une analyse complète, dont le mécanisme est pourtant simple. Edgar Poe (*Double assassinat dans la rue Morgue*) pressentait peut-être le théorème de Lucas lorsqu'il parlait (opinion que nous ne partageons aucunement) de "la laborieuse futilité des échecs" où "la complexité est prise – erreur fort commune – pour de la profondeur".

LE MINIMAX

Pour contourner cet énorme écueil, quelle stratégie pouvons-nous employer ? Hé bien, l'une des seules possibles en présence d'un système complexe et intégré : le transformer en système presque hiérarchisé par éclatement des objectifs. En clair, nous définirons, à partir de notre connaissance du jeu, une fonction d'évaluation, calculée du point de vue du programme, mais identique pour les deux adversaires. Elle valorisera toute position, notamment à une profondeur définie à partir de la position actuelle par la donnée d'un horizon H exprimé en nombre de niveaux. Nous admettrons que le jeu de l'adversaire est le meilleur possible, contre le programme, dans le contexte ainsi défini. L'opposant choisira donc de minimiser la fonction d'évaluation, tandis que notre programme la maximisera. Le raisonnement du paragraphe précédent, généralisé à toutes fonctions d'évaluation et non plus seulement à celles qui prennent les valeurs −1, 0, +1, permettra, par remontée dans l'arbre de jeu à partir de l'horizon, la valorisation de la position actuelle et le choix du meilleur coup.

Le programme MINIMAX va réaliser ce projet. Choisissons d'abord un exemple à titre de support. Le voici :



Les sommets de l'arbre figuré à 3 niveaux sont numérotés de 0 (la racine) à 27. Les chiffres en gras définissent les valeurs de la fonction d'évaluation pour les sommets du niveau 3. L'ordinateur joue le premier à partir de la racine. MINIMAX calcule, pour n'importe quel arbre de ce type, représenté en DATA :

- le coup optimal (variable SOPT),
- la valeur de la position actuelle (variable EVO),
- le nombre de coups simulés (variable CS).

Pour l'exemple choisi, après lancement du programme, l'écran affiche :

```
                    MINIMAX

 Ce programme calcule
la valeur du jeu et le coup optimaux
sur un arbre a 3 niveaux
defini par des DATA.

 Coup optimal: 1
 Valeur optimale: 4
 Nombre de coups simules: 27
```

Le cœur de MINIMAX est le sous-programme OPTIMUM listé ci-dessous.

```
230 '
240 '          OPTIMUM
250 '
260 ev0=-30000:cs=0
270 s0=ra:ns1=ns(ra)
280 FOR j1=1 TO ns1
290   ev1=30000:cs=cs+1
300   s1=su(s0,j1):ns2=ns(s1)
310   FOR j2=1 TO ns2
320     ev2=-30000:cs=cs+1
330     s2=su(s1,j2):ns3=ns(s2)
340     FOR j3=1 TO ns3
350       s3=su(s2,j3):ev3=va(s3):cs=cs+
1
360       ev2=MAX(ev2,ev3)
```

```
370      NEXT j3
380      ev1=MIN(ev1,ev2)
390    NEXT j2
400    IF ev1>ev0 THEN ev0=ev1:sopt=s1
410 NEXT j1
420 RETURN
```

Il offre une structure de boucles imbriquées identique à celle du sous-programme MAXIMUM de JEU1 et utilise les mêmes variables. Les modifications par rapport à JEU1 concernent les calculs successifs de maxima et de minima. D'où les signes alternés des valeurs initiales EV0,EV1, etc. et la succession des opérateurs MAX et MIN. L'horizon choisi correspond à trois niveaux mais toute autre valeur pourrait être adoptée avec des modifications évidentes.

Lors des premières opérations, le programme descend, si j'ose dire, le long de l'arbre (sommets 0, 1, 4, 12). La troisième boucle (indice J3 prenant la valeur 1) mémorise dans EV3 la valeur −2 du sommet numéro 12 et reporte cette valeur en EV2, par l'opération MAX(−30000, −2). Un deuxième tour de boucle (J3=2) stocke en EV3 la valeur 8 du sommet 13 et affecte à EV2 la valeur 8 par MAX(−2,8).

A la sortie de la troisième boucle (instruction 380), la valorisation EV1 du niveau 1 est recalculée par une opération de minimum, associée au jeu de l'adversaire :

EV1=MIN(30000,8)=8

Après remontée au sommet 1, le programme glisse vers le deuxième successeur (J2=2) de ce sommet et redonne à EV2 la valeur initiale −30000 pour assurer la correction des maximisations ultérieures faites dans la boucle 3 sur les sommets 14 et 15. La valeur maximale 4 de ces deux sommets est, comme plus haut, reportée en EV2 puis comparée à EV1=8. Le minimum :

MIN(8,4)=4

donne la nouvelle valeur de EV1. L'exploration du sommet 16 ne la modifie pas. Nous revenons alors à la première boucle, à la racine et au sommet 2, deuxième successeur de celle-ci (J1=2). Au passage, la racine est valorisée du maximum :

EV0=MAX(−30000,4)=4

En même temps, le meilleur coup provisoire (S1=1) qui a fourni ce maximum est mémorisé dans SOPT... et le petit jeu continue que vous pourrez, sans problème je suppose, suivre jusqu'au bout. Voici maintenant la liste complète du programme MINIMAX, précédée d'une liste des variables.

Liste des variables

Tables :

NS(NMAX)	: nombre des successeurs d'un sommet.
SU(NMAX, NSMAX)	: numéros repères des successeurs par sommet et par rang.
VA(NMAX)	: valorisations des sommets.

Variables :

CS	: compteur de sommets explorés.
EV0, EV1, EV2, EV3	: maxima à chaque niveau.
I, J, J1, J2, J3	: indices des boucles.
N	: numéros repères des sommets.
NMAX	: nombre maximal de sommets.
NSMAX	: nombre maximal de successeurs par sommet.
NS1, NS2, NS3	: nombres courants de successeurs à chaque niveau.
RA	: numéro repère de la racine.
S0, S1, S2, S3	: repères des sommets courants à chaque niveau.
SOPT	: coup (sommet) optimal à partir de la racine.

Listing du programme MINIMAX

```
10 MODE 1:INK 0,13:INK 1,0:PAPER 0:PEN 1
:CLS:PRINT:PRINT TAB(17)"MINIMAX":PRINT
20 PRINT " Ce programme calcule"
30 PRINT "la valeur du jeu et le coup op
timaux"
40 PRINT "sur un arbre a 3 niveaux"
50 PRINT "defini par des DATA.":PRINT
60 DEFINT a-z:nmax=30:nsmax=3
70 DIM ns(nmax),su(nmax,nsmax),va(nmax)
80 ra=0
90 DATA 3,3,2,3,2,2,1,2,2,2,3,2
100 FOR i=0 TO 11:READ ns(i):NEXT
110 FOR i=0 TO 11
120   FOR j=1 TO ns(i)
130     n=n+1:su(i,j)=n
140   NEXT j
150 NEXT i
160 DATA -2,8,3,4,15,-12,-6,8,-5,0,3,16,
2,-15,3,-1
170 FOR i=12 TO 27:READ va(i):NEXT
```

```
180 GOSUB 260:'Optimum
190 PRINT " Coup optimal:";sopt
200 PRINT " Valeur optimale:";ev0
210 PRINT " Nombre de coups simules:";cs
:PRINT
220 END
230 '
240 '        OPTIMUM
250 '
260 ev0=-30000:cs=0
270 s0=ra:ns1=ns(ra)
280 FOR j1=1 TO ns1
290   ev1=30000:cs=cs+1
300   s1=su(s0,j1):ns2=ns(s1)
310   FOR j2=1 TO ns2
320     ev2=-30000:cs=cs+1
330     s2=su(s1,j2):ns3=ns(s2)
340     FOR j3=1 TO ns3
350       s3=su(s2,j3):ev3=va(s3):cs=cs+
1
360       ev2=MAX(ev2,ev3)
370     NEXT j3
380     ev1=MIN(ev1,ev2)
390   NEXT j2
400   IF ev1>ev0 THEN ev0=ev1:sopt=s1
410 NEXT j1
420 RETURN
```

L'omega des programmes de jeu : l'Alpha-beta

Construire un programme convenable pour un jeu classique à deux personnes est, vous l'avez compris, une bataille contre le temps de calcul et la capacité de mémoire. L'algorithme du MINIMAX est une arme précieuse. Il faut, cependant, l'améliorer.

Exemple : un programme d'échecs performant doit conduire, au-delà de quatre niveaux, l'exploration de l'arbre de jeu. A 32 configurations, en moyenne, par niveau, quatre niveaux signifient 32^4, soit un million de positions. Or, sur micro-ordinateur, même en langage machine, il est difficile de dépasser une vitesse limite de traitement voisine de 500 à 700 coups par seconde. Faites le calcul du temps moyen de "réflexion" par coup : une demi-heure. C'est, en pratique, inacceptable.

A une date relativement récente (il y a vingt ans environ), plusieurs informaticiens firent une remarque importante. Il était possible, dans l'application du MINIMAX, d'élaguer fortement l'arbre de jeu sans réduire les performances.

Soit un sommet S, au niveau N pair, affecté de la valorisation V au cours du processus d'exploration de gauche à droite. Par convention, la

position actuelle du jeu, racine de l'arbre, est au niveau zéro et le programme doit jouer. Aux niveaux pairs, l'algorithme cherche donc un maximum.

Considérons un successeur S' du sommet S (jeu du programme) et un "petit-fils" S'' au niveau N+2 (jeu de l'adversaire). Supposons calculée, à cet instant, une valorisation V" de S". Si :

V"<=V

alors **il est inutile de poursuivre l'évaluation des successeurs du sommet S'**. Nous pouvons remonter à S et changer de branche. L'explication est simple : si le programme choisissait d'aller en S', l'adversaire, jouant vers S", pourrait lui imposer la valorisation V", inférieure ou au plus égale à la valeur V que le programme est sûr d'obtenir selon les analyses déjà faites. Cette conclusion est valable quelles que soient les valorisations des autres successeurs de S'. La branche S' n'offre pas d'intérêt. Nous pouvons, sans regret, passer à un autre successeur de S.

Pour des raisons analogues, si N est un nombre impair (l'adversaire joue à partir de la position S), la condition :

V">=V

permet d'écarter la branche S' que l'adversaire, supposé compétent et informé, ne retiendra pas. S'il le faisait, le programme pourrait lui imposer la valorisation V" supérieure ou égale à la valeur V déjà réalisable par d'autres choix.

Reprenons l'arbre de jeu qui nous a permis l'essai de MINIMAX. La branche de gauche (vers le sommet 1) a déjà été explorée. Elle donne à la racine (sommet S) la valeur provisoire V=4. A partir des sommets terminaux 17 et 18, le sommet 7 (noté S") est valorisé à V"=MAX(−12,−6)=−6. Cette valeur est inférieure à V. La seconde branche à partir de la racine (vers le sommet 2) s'avère donc sans intérêt. Il est inutile d'explorer les sommets 8, 19 et 20. Il convient de sauter à la troisième branche (sommet 3).

Pour celle-ci, après valorisation du sommet 9 (V"=3), la même conclusion s'impose : cette branche doit être écartée. Nous économisons ainsi les sommets 23 à 27.

Le programme ALBE

Il suffira de peu de chose pour transformer MINIMAX en un programme plus performant, qui exploite l'algorithme alpha-beta. Après compression (les noms de programmes doivent comporter 8 lettres au plus), nous l'appellerons ALBE, ce qui est plus joli. ALBE diffère de MINIMAX par deux instructions seulement, aux lignes 350 et 380 du listing joint. Elles traduisent, sans plus, les conditions que nous venons de reconnaître.

Nos deux derniers programmes contiennent une variable CS (compteur de sommets) non évoquée jusqu'ici. Nous l'avons introduite précisément pour mesurer le gain qu'apporte l'alpha-beta. Si nous lançons le programme ALBE, l'écran devrait présenter l'aspect suivant :

```
                    ALPHA-BETA

 Ce programme calcule
la valeur du jeu et le coup optimaux
sur un arbre a 3 niveaux
defini par des DATA.

 Coup optimal: 1
 Valeur optimale: 4
 Nombre de coups simules: 17
```

Ainsi, nous avons, pour le même résultat, simulé 17 coups au lieu de 27, soit, sur cet exemple, une économie de 37 %.

```
10 MODE 1:INK 0,13:INK 1,0:PAPER 0:PEN 1
:CLS:PRINT:PRINT TAB(15)"ALPHA-BETA":PRI
NT
20 PRINT " Ce programme calcule"
30 PRINT "la valeur du jeu et le coup op
timaux"
40 PRINT "sur un arbre a 3 niveaux"
50 PRINT "defini par des DATA.":PRINT
60 DEFINT a-z:nmax=30:nsmax=3
70 DIM ns(nmax),su(nmax,nsmax),va(nmax)
80 ra=0
90 DATA 3,3,2,3,2,2,1,2,2,2,3,2
100 FOR i=0 TO 11:READ ns(i):NEXT
110 FOR i=0 TO 11
120   FOR j=1 TO ns(i)
130     n=n+1:su(i,j)=n
140   NEXT j
150 NEXT i
160 DATA -2,8,3,4,15,-12,-6,8,-5,0,3,16,
2,-15,3,-1
170 FOR i=12 TO 27:READ va(i):NEXT
180 GOSUB 260:'Alpha-beta 3 niveaux
190 PRINT " Coup optimal:";sopt
```

```
200 PRINT " Valeur optimale:";ev0
210 PRINT " Nombre de coups simules:";cs
:PRINT
220 END
230 '
240 '        Alpha-beta 3 niveaux
250 '
260 ev0=-30000:s0=ra:ns1=ns(ra):cs=0
270 FOR j1=1 TO ns1
280   ev1=30000:cs=cs+1
290   s1=su(s0,j1):ns2=ns(s1)
300   FOR j2=1 TO ns2
310     ev2=-30000:cs=cs+1
320     s2=su(s1,j2):ns3=ns(s2)
330     FOR j3=1 TO ns3
340       s3=su(s2,j3):ev3=va(s3):cs=cs+
1
350       IF ev3>=ev1 THEN 400
360       ev2=MAX(ev2,ev3)
370     NEXT j3
380     IF ev2<=ev0 THEN 420
390     ev1=MIN(ev1,ev2)
400   NEXT j2
410   IF ev1>ev0 THEN ev0=ev1:sopt=s1
420 NEXT j1
430 RETURN
```

Pour aller plus loin

L'algorithme Alpha-beta élague les branches de l'arbre qui conduiraient à des résultats inférieurs à ceux déjà obtenus au cours de l'exploration. D'où l'avantage de commencer par les meilleurs coups et donc de ranger ceux-ci par intérêt décroissant.

Selon de récentes recherches, si l'exploration de l'arbre de jeu est effectuée dans l'ordre optimal, le nombre de nœuds examinés avoisinera $2*\sqrt{N}$, où N est le nombre total de nœuds terminaux. A titre d'exemple, si le facteur de croissance (nombre moyen de successeurs par sommet) est 16 et l'horizon égal à 3 niveaux, les performances, en nombre moyen de coups terminaux examinés, peuvent être ainsi comparées :

Nombre total :	4096
Alpha-beta au hasard :	1214
Alpha-beta ordonné :	128

Ainsi, dans le deuxième cas, 30 % des sommets terminaux seront examinés (gain de 70 %) et dans le troisième cas 3 % (gain de 97 %). Ces chiffres illustrent la puissance de l'Alpha-beta et l'intérêt d'un tri préalable des coups. Dans l'hypothèse courante d'une fonction d'éva-

luation définie pour chaque sommet, cette dernière opération peut être réalisée de deux façons :

- utiliser comme critère de tri la valeur de la fonction d'évaluation au premier niveau (coups directs à partir de la racine) ;
- procéder à une "recherche courte" préalable sur un petit nombre de niveaux afin d'ordonner les coups lors d'une exploration plus profonde.

Le problème essentiel, nous l'avons dit, est de bloquer l'explosion combinatoire et, par conséquent, d'élaguer, mais à bon escient, l'arbre de jeu. L'Alpha-beta est une méthode efficace et générale mais beaucoup d'autres voies, alternatives ou complémentaires, sont ouvertes. Les décrire en détail déborderait le cadre de ce livre. Bornons-nous à citer quelques heuristiques importantes :

a) Le "coup meurtrier"

Lorsque l'algorithme provoque un saut de branche (avec élagage), cela signifie la rencontre d'un coup efficace du point de vue de l'adversaire. Ce coup réapparaîtra sans doute, au même niveau, pour d'autres positions. Il est donc important de le noter et de l'essayer systématiquement en première position, au niveau considéré, avec une probabilité accrue d'autres élagages.

b) La sélection *a priori*

Il est parfois possible de définir des critères de priorité ou, au contraire, d'élagage, critères indépendants de la fonction d'évaluation. Ils permettent une sélection préalable des coups et contribuent à la réduction de l'arbre. Le choix de ces critères pose cependant de considérables problèmes.

c) L'horizon variable

Certaines positions de jeu sont "calmes". D'autres déclenchent, aux échecs par exemple, des échanges de pièces dont il convient de prévoir les lointaines conséquences. L'horizon d'exploration doit donc être ajustable selon les circonstances rencontrées.

d) Les systèmes experts

La fonction d'évaluation est une manière de traduire le savoir des joueurs expérimentés. L'expérience de ceux-ci peut, également, s'exprimer en règles d'action qui associent des choix définis à la rencontre de certaines situations (*cf.* troisième partie de cet ouvrage). Cette méthode, encore très peu utilisée, est, sans doute, appelée à un bel avenir.

Chapitre 6
DES CAILLOUX ET DES HOMMES

Nous voulons, dans ce chapitre, créer un ou plusieurs programmes qui puissent tenir leur place à un très bon niveau, dans un jeu stratégique connu, à deux personnes. Pour les raisons développées ci-dessous, nous avons choisi l'awele, qui appartient à la nombreuse famille des Wari-solo.

WARI ET SOLO

Ce sont des "jeux de calculs" dans tous les sens du mot (opérations arithmétiques et cailloux). Ils utilisent des tabliers, généralement en bois, creusés de petites cases (ou bols) pouvant contenir cailloux, graines, pions ou haricots en nombres donnés au départ. Ces tabliers sont partagés, dans le sens de la longueur, en deux parties, affectées chacune à un joueur.

Principe général : chaque joueur, à tour de rôle, choisit une des cases de la rangée lui appartenant, prend dans sa main les cailloux qui s'y trouvent et les sème, un par un, dans les cases suivantes en sens inverse des aiguilles d'une montre. La suite dépend du contenu de la case dans laquelle tombe le dernier caillou et donne lieu à une très grande variété de règles et de jeux. Voici, par exemple, les règles du kalah)*cf. Wari et solo* par A. Deledicq et A. Popova 1977, Cedic Editeur, page 13).

Le tablier comporte deux rangées de 7 cases : 6 petites cases et une grande case à droite (le kalah), pour chaque camp. Au départ, chacune des petites cases contient 3 pions, soit 36 au total. Lors de chaque "semis", un pion est laissé dans la case de départ. La grande case à la droite de chaque joueur bénéficie de la distribution et les pions qui y sont laissés au passage constitueront le gain des joueurs en fin de partie. On ne sème pas dans la case de droite de son adversaire.

Si le dernier pion tombe dans la case de droite du joueur qui opère, celui-ci obtient le droit de rejouer. Si ce même dernier pion tombe dans une des cases vides du joueur, face à une case pleine de l'adversaire, le joueur prend les pions de cette case adverse, y ajoute le dernier pion et place le tout dans son kalah. Lorsqu'un joueur n'a plus de pions, son adversaire dépose les siens dans son propre kalah. Le vainqueur est celui qui a accumulé le plus de pions dans sa grande case.

Les variétés de Wari-solo (il en existerait plus d'un millier) se pratiquent dans toute l'Afrique, l'Asie du Sud et une partie de l'Asie Centrale. Elles se distinguent des jeux occidentaux sur un point essentiel : dans ces derniers, les pions ou pièces sont la priorité de chaque joueur et luttent sur un champ de bataille à conquérir. Au contraire, les joueurs de Wari délimitent et possèdent chacun leur propre territoire, tandis qu'ils échangent constamment des "matériels" (pions) de couleur et de forme indifférenciées. De ce point de vue, Othello est un jeu mixte. Je vous laisse rêver aux correspondances culturelles qu'il vous plairait d'imaginer.

Pourquoi choisir de tels jeux au lieu de nos bien classiques échecs ou dames ? Nous voulons, je le rappelle, réaliser des programmes qui puissent jouer à un excellent niveau. Mais ils doivent être écrits en BASIC, exécutés sur micro-ordinateur personnel et se prêter à un jeu agréable, ce qui impose, en particulier, des "temps de réflexion" courts. Compte tenu des facteurs de croissance (nombre de possibilités à chaque coup), de l'ordre de 30 ou 40 pour dames et échecs, ces conditions ne peuvent être approchées qu'en langage d'assembleur.

Dans les Wari, au contraire, le facteur de croissance est beaucoup plus faible (6 au maximum pour l'awele qui sera notre choix). Les règles sont simples ; la représentation informatique du jeu peut l'être également. Le tablier est aisément figuré sans longue liste d'instructions. Il s'agit pourtant de jeux passionnants, raffinés et d'une grande profondeur stratégique : les incidences de chaque coup peuvent se propager jusqu'à 10 ou 12 niveaux de l'arbre de jeu. Les coups décisifs se préparent de longue date.

Les règles de l'awele

Nous bâtirons nos programmes à partir de ce jeu, dont voici les règles.

Le tablier abrite 2 rangées de 6 cases. Chacune est le territoire d'un joueur. Au départ, chaque case contient 4 pions, donc 48 au total.

A chaque coup, le joueur concerné choisit une case non vide de son camp, en enlève les pions et les sème un à un dans les cases suivantes, en sens inverse des aiguilles d'une montre.

Si le dernier pion tombe dans une **case adverse contenant 1 ou 2 pions**, le joueur prend le contenu de cette case, y compris le dernier

pion qu'il vient d'y semer. Si les cases adverses précédentes (dans l'ordre du semis) contiennent aussi 2 ou 3 pions à l'issue du coup (elles en renfermaient donc 1 ou 2 au départ) et forment avec la case d'arrivée une chaîne ininterrompue, elles sont également vidées au profit du joueur et les pions correspondants sont sortis du jeu. Ces prises sont définitives.

Si le joueur, au cours d'un semis, fait un tour complet du tablier, il doit sauter sa case de départ.

Enfin, aucun joueur ne doit priver son adversaire de la possibilité de jouer en gardant toutes les graines de son propre côté. Cela s'appelle "lui donner à manger". En cas d'infraction à cette règle, l'adversaire ramasse tous les pions sur le tablier et les met dans son tas, achevant ainsi la partie.

Est déclaré gagnant le joueur qui s'est approprié plus de la moitié des pions, soit 25 ou davantage. La partie peut être déclarée nulle, d'un commun accord, s'il reste si peu de pions sur le tablier qu'ils deviennent inaccessibles les uns pour les autres.

CRÉATION D'UN PROGRAMME DE JEU

La structure générale d'un programme jouant à l'une des variétés de Wari pourra être définie comme suit :

IN) – **Initialisation** (titre, dimension des tables, etc.)
Construction du cadre de jeu à l'écran
Mise à jour et affichage des positions initiales
Interrogation : qui joue le premier ?
Si joueur humain ALORS aller en JH
SINON aller en JO

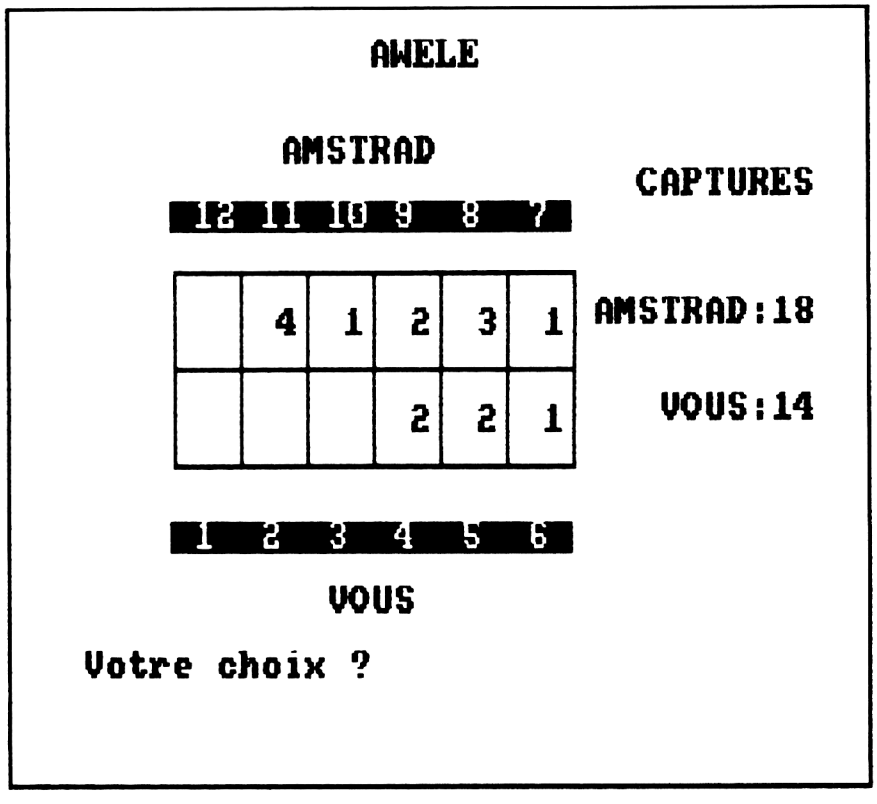
JH) – **Jeu du partenaire humain**
Vérifier qu'une case, au moins, n'est pas vide
Choix du coup
Contrôle de validité
Traitement du coup (transferts entre bols et calcul des captures)
Affichage des positions
Vérifier les conditions de fin de partie :
Si elles sont satisfaites Alors aller en FP
SINON aller en JO

JO) – **Jeu de l'ordinateur**
Vérifier qu'une case, au moins, n'est pas vide
Exploration de l'arbre de jeu jusqu'à l'horizon choisi, avec évaluation des coups terminaux et remontée vers la racine
Détermination du coup optimal

Traitement du coup (transferts entre bols et calcul des captures)
Affichage des positions
Vérifier les conditions de fin de partie :
Si elles sont satisfaites ALORS aller en FP
SINON aller en JH

FP) – **Fin de partie**
La partie est-elle nulle ? Sinon quel est le gagnant ?
Afficher, dans chaque cas, le score et les messages utiles
Demander si l'on veut rejouer
Si oui ALORS aller en IN
SINON fin

Avant d'aborder l'analyse détaillée des programmes, nous devons choisir la représentation informatique des positions et le mode d'affichage du jeu à l'écran. Pour le premier point, une table de nombres entiers, BOL(14) de dimension 14, fera très bien l'affaire. Les douze premiers éléments (indices 1 à 12) mémoriseront les nombres de pions dans chacune des douze cases, les six premières réservées au joueur humain et les six autres à l'ordinateur. L'élément d'indice 13 contiendra le nombre de pions capturés par le joueur humain et celui d'indice 14 les prises de l'Amstrad. La copie d'écran ci-dessous indique la configuration du tableau de jeu.



Pour une approche graduée, nous bâtirons successivement deux programmes. Le premier, AWELE1, se contentera d'une exploration au premier niveau (coups directs possibles de l'ordinateur à partir de la position actuelle). Le second, AWELE2, conduira une recherche sur trois niveaux.

POUR VOS DÉBUTS, VOICI AWELE1

Le terrain de jeu

Suivons le plan de construction du programme. Nous devons dessiner, à l'écran, le cadre de jeu et afficher les positions initiales. Deux sous-programmes se chargeront de ces fonctions.

Le premier (instructions 520 à 720), que voici, place les titres et libellés, dessine le tablier, numérote les cases.

Liste des variables

DEL : dimension du côté des carrés formant le tablier (48 pixels).

X0, Y0 : coordonnées du coin inférieur gauche du tablier, soit X0= 130 pixels et Y0= 170 pixels.

X,Y : numéros des cases du tablier, en largeur et hauteur.

```
520 '
530 '          CADRE DU JEU
540 '
550 LOCATE 14,5:PRINT "AMSTRAD"
560 LOCATE 30,6:PRINT "CAPTURES"
570 LOCATE 28,10:PRINT "AMSTRAD:":LOCATE
 31,13:PRINT "VOUS:"
580 LOCATE 16,19:PRINT "VOUS"
590 ORIGIN x0,y0
600 FOR y=0 TO 2
610   PLOT 0,y*del:DRAWR 288,0
620 NEXT y
630 FOR x=0 TO 6
640   PLOT x*del,0:DRAWR 0,96
650 NEXT x
660 PAPER 1:PEN 0
670 FOR x=0 TO 5
680   LOCATE 9+x*3,7:PRINT 12-x
```

```
690    LOCATE 9+x*3,17:PRINT 1+x
700 NEXT x
710 PAPER 0:PEN 1
720 RETURN
```

L'affichage d'une position quelconque est réalisé par un second sous-programme (instructions 980 à 1130) à partir de la table BOL (14) qui mémorise cette position.

Liste des variables

BOL(14) : table des positions et captures.
BOL$(14) : même table en format modifié, pour l'impression.
CAPTJ : indice de l'élément de la table BOL() contenant les captures du joueur (CAPTJ= 13).
CAPTOR : indice correspondant pour les captures de l'ordinateur.
I : indice de boucle.
NS : nombre maximal de coups possibles (successeurs) à partir d'une position donnée (NS= 6).

```
980 '
990 '          AFFICHAGE DES POSITIONS
1000 '
1010 TAG
1020 ORIGIN x0+0.2*del,y0+0.65*del
1030 FOR i=1 TO captor
1040    IF bol(i)=0 THEN bol$(i)="  " ELS
E bol$(i)=STR$(bol(i)):bol$(i)=MID$(bol$
(i),2):IF LEN(bol$(i))=1 THEN bol$(i)="
"+bol$(i)
1050 NEXT i
1060 FOR i=1 TO ns
1070    MOVE (i-1)*del,0:PRINT bol$(i);
1080    MOVE (i-1)*del,del:PRINT bol$(13-
i);
1090 NEXT i
1100 TAGOFF
1110 LOCATE 36,10:PRINT bol$(captor)
1120 LOCATE 36,13:PRINT bol$(captj)
1130 RETURN
```

La transposition de BOL() en BOL$() a pour but :

– d'éliminer le signe ou espace qui, dans la représentation interne, précède tout nombre ;

– de remplacer le nombre zéro par deux espaces ;
– de cadrer à droite les nombres affichés.

Le générateur de coups

A chaque phase du jeu, à partir d'une position (tablier et captures) définie par la table BOL(), le joueur humain ou l'ordinateur choisissent un coup par son numéro de 1 à 6 (variable COUP). Le générateur de coups doit accepter ces données et renvoyer :

– la position du jeu qui en résulte (configuration fille du graphe) ;
– les captures éventuelles réalisées par le joueur humain ou l'ordinateur.

Il faut encore mettre à jour le tableau des captures et appeler le sous-programme d'affichage. Les sous-programmes JEU et TRAITEMENT D'UN COUP, ci-dessous listés, assurent ces fonctions.

Liste des variables

CAPH : nombre de captures du joueur humain lors du coup joué.
CAPO : nombre de captures de l'ordinateur dans les mêmes conditions.
CF(NBOL) : table intermédiaire mémorisant le contenu des 12 cases du tablier.
COUP : numéro du coup choisi (de 1 à 6).
JO$: chaîne de caractères désignant le joueur qui opère. Elle peut prendre les deux valeurs "hum" pour le joueur humain et "ord" pour l'ordinateur.
NBOL : nombre de cases du tablier (NBOL= 12).
NC : nombre de pions dans la case jouée.
X,Y : variables intermédiaires désignant les numéros des cases à mesure de leur traitement.

```
730 '
740 '          JEU
750 '
760 FOR i=1 TO nbol:cf(i)=bol(i):NEXT
770 GOSUB 850:' Traitement d'un coup
780 FOR i=1 TO nbol:bol(i)=cf(i):NEXT
790 bol(captj)=bol(captj)+caph:bol(capto
r)=bol(captor)+capo
800 GOSUB 1010:' Affichage des positions
810 RETURN
820 '
830 '          TRAITEMENT D'UN COUP
```

```
840 '
850 IF jo$="ord" THEN coup=coup+6
860 nc=cf(coup):cf(coup)=0:y=coup
870 FOR x=1 TO nc
880   y=y MOD 12+1:IF y=coup THEN 880
890   cf(y)=cf(y)+1
900 NEXT
910 caph=0:capo=0
920 IF (y-6.5)*(coup-6.5)>0 THEN RETURN
930 WHILE (y-6.5)*(coup-6.5)<0 AND (cf(y
)=2 OR cf(y)=3)
940   IF coup<=6 THEN caph=caph+cf(y) EL
SE capo=capo+cf(y)
950   cf(y)=0:y=y-1:IF y=0 THEN y=12
960 WEND
970 RETURN
```

Je vous suggère d'analyser avec attention le sous-programme TRAITEMENT D'UN COUP. Il constitue, avec EXPLORATION, dont nous ferons tout à l'heure connaissance, le cœur des programmes de ce chapitre. Le BASIC structuré de l'Amstrad permet ici une écriture dense et, je l'espère, claire.

La boucle (lignes 870 à 900) simule l'opération de semis avec ses particularités : cheminement circulaire dans le tablier (opérateur MOD), saut de la case de départ si elle est rencontrée, dépôt d'un pion dans chaque case.

Les instructions 910 à 960 concernent les opérations de capture. Celles-ci partent de la case d'arrivée si elle se trouve chez l'adversaire, ce qu'exprime, dans tous les cas, la condition :

$$(Y-6.5)*(COUP-6.5)< 0$$

Les captures intéressent, selon les règles du jeu rappelées au début de ce chapitre, la case adverse d'arrivée qui, après le jeu du coup, contient 2 ou 3 pions et toutes les cases adverses qui la précèdent et vérifient, sans interruption, la même condition.

Analyse d'une position : menaces et maisons

Pour définir une fonction d'évaluation, nous devons reconnaître les forces et faiblesses d'une position et leur associer des indicateurs.

Quelle est la finalité du jeu ? Capturer, au cours d'une partie, plus de pions adverses que vous n'en perdrez vous-même. Or une case Y est menacée de capture si :

– elle contient un ou deux pions ;

– elle est sous la visée directe d'une case X de l'adversaire et donc est une case terminale d'un semis partant de X.

Cette dernière condition s'exprime ainsi :

X+contenu de X=Y modulo 12

Par ailleurs, une suite ininterrompue de cases à 1 ou 2 pions est très vulnérable (captures en chaîne). Soit, à titre d'exemple, la configuration suivante (c'est l'Amstrad qui joue) :

Amstrad

12	11	10	9	8	7
6		1	2	1	
				4	2
1	2	3	4	5	6

Vous

La case 12 exerce une menace sur la case 6. Si l'ordinateur joue la case 12, il capturera 3 pions. Mais les cases 9 et 8 sont menacées par votre case 5 et la case 8 par la case 6. Par ailleurs, si l'Amstrad joue la case 12, au coup suivant, la case 5 contiendra 5 pions. En jouant à votre tour cette case qui menacera la case 10, vous raflerez 8 pions ! L'ordinateur serait donc plus avisé de jouer la case 9, pour contrer l'attaque à partir de la case 5 ou, mieux encore, la case 8 qui pare à toutes les menaces.

Autre notion importante dans les jeux de type Wari : la construction d'une "maison". On appelle ainsi tout groupe de 12 pions au moins dans une même case. Étudions l'exemple suivant où l'ordinateur doit jouer et voyons ce qui se passe s'il joue la case 11.

Amstrad

12	11	10	9	8	7
	16		2		
3					
1	2	3	4	5	6

Vous

Après le coup et avant retrait des pions capturés, le tablier offre l'aspect suivant, selon les règles du jeu (saut de la case départ quand elle est rencontrée au cours du semis) :

Amstrad

12	11	10	9	8	7
2		1	3	1	1
5	2	2	2	1	1
1	2	3	4	5	6

Vous

Ainsi, la case terminale est le 4 et le nombre de captures égal à 6. Construire une maison mais, surtout, la détruire au moment opportun peut donc être une opération très avantageuse. Elle le sera d'autant plus que la maison se situera vers la droite du joueur et qu'elle visera une case adverse vide ou contenant 1 pion au plus.

La leçon est claire et, d'ailleurs, conforme à l'intuition. La recherche du meilleur coup à partir d'une position donnée ne doit pas se limiter au nombre de pions capturés pour chaque coup mais prendre en compte, à travers des critères à définir, le "potentiel" de la configuration et donc sa capacité de gains ou pertes ultérieurs.

Fonctions d'évaluation

Le nombre de captures associées au jeu d'un coup est, bien sûr, l'élément le plus important d'une fonction d'évaluation. Reste à estimer le potentiel (captures ultérieures espérées) de la configuration qui succédera à ce coup. Ce potentiel croît avec :

- le nombre total de pions sur le territoire du joueur ;
- la concentration de ces pions liée, par exemple, au nombre maximal de pions sur une case du joueur ;
- la position de ce maximum : une maison doit, en effet, viser une case adverse après un tour complet du tablier ; c'est mieux encore si cette case adverse est vide ou contient un seul pion ;
- le nombre de cases adverses contenant au plus 2 pions.

Ce même potentiel décroît avec le nombre de cases du joueur contenant au plus 2 pions.

Autres caractéristiques pouvant intervenir utilement :

- le nombre de menaces sur les cases adverses ;

– le nombre de "pièges" (visées sur la case adverse vide qui suit une case pleine).

Chaque critère peut, dans la fonction d'évaluation, apparaître seul (valorisé pour un joueur) ou comme différence entre les évaluations de même type relatives aux deux camps. Exemple : différence des sommes de pions de chaque côté du tablier.

Nous sommes donc libres d'imaginer et d'essayer (il s'agit bien d'heuristiques) une grande variété de critères. Mais les difficultés croissent rapidement avec la complexité :

– il faut estimer, essayer, réévaluer les coefficients de pondération associés aux divers facteurs (voir chapitre suivant) ;
– les temps de calcul peuvent devenir prohibitifs.

Nous tenterons, pour commencer, l'approche la plus simple, définie par les deux sous-programmes EXPLORATION et EVALUATION que voici, précédés de la liste des variables complémentaires utilisées.

Liste des variables

C	: numéro courant d'une case.
COUOPT	: numéro du meilleur coup.
E1, E2	: coefficients de la fonction d'évaluation.
EV0, EV1	: évaluations courantes de la position initiale et d'une position fille.
SOM	: total du contenu des cases côté ordinateur.
SOMJH	: total du contenu des cases côté joueur humain.
SUP	: valeur maximale des cases côté ordinateur.

```
1140 '
1150 '          EXPLORATION
1160 '
1170 ev0=-30000
1180 FOR c=1 TO ns
1190   FOR i=1 TO nbol:cf(i)=bol(i):NEXT
1200   IF cf(c+6)=0 THEN 1230
1210   coup=c:GOSUB 850:GOSUB 1280
1220   IF ev1>ev0 THEN ev0=ev1:coupopt=c
1230 NEXT
1240 RETURN
1250 '
1260 '          EVALUATION
1270 '
1280 somjh=0:FOR j=1 TO ns:somjh=somjh+c
f(j):NEXT
1290 som=0:sup=0
1300 FOR j=ns+1 TO 2*ns
```

```
1310    sup=MAX(sup,cf(j)):som=som+cf(j)
1320 NEXT
1330 ev1=capo+e1*sup+e2*som
1340 IF somjh=0 THEN ev1=ev1-som
1350 RETURN
```

Le sous-programme EXPLORATION simule chaque coup jouable par l'Amstrad à partir de la position actuelle et donc, à cette fin :

- transfère le contenu des cases du tablier (table BOL(I)) dans une table intermédiaire CF(I) ;
- joue chaque coup, de numéro C, en appelant le sous-programme TRAITEMENT D'UN COUP ; la position résultante est mémorisée dans CF(I) ;
- appelle EVALUATION pour valoriser cette position fille ;
- détermine le maximum de ces valorisations et le meilleur coup qui lui est associé.

La fonction d'évaluation est la somme des termes suivants :

- nombre de captures (soit CAPO) obtenues par l'ordinateur en jouant le coup ;
- produit par le coefficient EV1 de la valeur maximale (SUP) du contenu des cases de la position fille côté ordinateur ;
- produit par le coefficient EV2 de la somme (SOM) du contenu des cases de la position fille côté ordinateur.

Souvenons-nous enfin de la règle "donner à manger" et des sanctions associées. Si l'adversaire, après un coup de votre part, ne peut jouer faute de pions, il ramassera à son profit tous ceux du tablier, c'est-à-dire tous les vôtres. La fonction d'évaluation en tient compte : elle réduit, dans ce cas, la valorisation du coup du contenu total des cases, côté ordinateur.

Nous avons donné des valeurs vraisemblables (0,15 et 0,08) aux coefficients de pondération EV1 et EV2. Selon ces valeurs, une "maison" de 20 pions signifie un potentiel de captures égal à :

$$(EV1+EV2)*20=4,6$$

Le chapitre suivant sera plus explicite quant aux méthodes d'estimation de ces coefficients.

Le programme AWELE 1

Le corps du programme (instructions 10 à 510) est simple et n'appelle pas de commentaires. Après les diverses initialisations, la boucle principale passe la main tantôt à l'ordinateur, tantôt au joueur humain, jusqu'à la fin de la partie. Vous le verrez en essayant AWELE1 dont voici le listing et les variables regroupées.

Liste des variables

Tables :

BOL(CAPTOR) : table des positions et captures.
BOL$(CAPTOR) : même table en format modifié.
CF(NBOL) : table intermédiaire mémorisant les contenus des 12 cases du tablier.

Variables :

C : numéro courant d'une case.
CAPH : capture par le joueur humain en un coup.
CAPO : captures par l'ordinateur en un coup.
CAPTJ : captures cumulées du joueur humain.
CAPTOR : captures cumulées de l'ordinateur.
COUP : numéro du coup joué (entre 1 et 6).
COUPOPT : numéro du coup optimal.
DEL : dimension du côté des carrés formant le tablier (48 pixels).
ECART : différence entre les captures cumulées de l'ordinateur et du joueur.
E1, E2 : coefficients de la fonction d'évaluation.
EV0, EV1 : évaluations courantes de la position initiale et d'une position fille.
I, J : indices de boucles.
JO$: chaîne de caractères désignant le joueur qui opère.
LI : numéro d'une ligne d'écran.
NBOL : nombre de cases du tablier (NBOL=12).
NC : nombre de cases du tablier (NBOL=12).
NS : nombre de pions dans la case jouée.
R : nombre maximal de coups possibles (NS=6).
R$: numéro du coup choisi par le joueur humain.
SOM : chaîne de caractères associée à R.
SOMJH : total du contenu des cases côté ordinateur.
SUP : total du contenu des cases côté joueur humain.
TITRE$: valeur maximale des cases côté ordinateur.
X, Y : titre du programme.
X0, Y0 : numéros des cases du tablier, en largeur et hauteur.
: coordonnées du coin inférieur gauche du tablier.

Listing du programme AWELE1

```
10 DEFINT a-d,f-z:titre$="AWELE1"
20 MODE 1:INK 0,1:INK 1,24:PAPER 0:PEN 1
30 CLS:PRINT:PRINT TAB(20-LEN(titre$)/2)
;titre$:PRINT
```

```
40 PLOT 10,10:DRAW 630,10:DRAW 630,398:D
RAW 10,398:DRAW 10,10
50 nbol=12:captj=13:captor=14:ns=6:x0=13
0:y0=170:del=48:e1=0.15:e2=0.08
60 DIM bol(captor),bol$(captor),cf(nbol)
70 GOSUB 550:' Cadre du jeu
80 FOR i=1 TO nbol:bol(i)=4:NEXT
90 bol(captj)=0:bol(captor)=0
100 GOSUB 1010:' Affichage des positions
110 li=21:LOCATE 5,li:PRINT "Voulez vous
 jouer le premier ?";:r$=""
120 WHILE r$<>"n" AND r$<>"o"
130   r$=INKEY$:r$=LOWER$(r$)
140 WEND
150 PRINT r$:GOSUB 1390:' Effacement d'u
ne ligne
160 IF r$="n" THEN 330
170 '
180 ' Joueur humain
190 '
200 som=0:FOR i=1 TO ns:som=som+bol(i):N
EXT
210 IF som=0 THEN FOR i=ns+1 TO 2*ns:bol
(captj)=bol(captj)+bol(i):bol(i)=0:NEXT:
GOSUB 1010:GOTO 440
220 li=22:GOSUB 1390:' Effacement d'une
ligne
230 LOCATE 5,li:PRINT "Votre choix ?";:r
$="":r=0
240 WHILE r<1 OR r>6 OR bol(r)=0
250   r$=INKEY$:r=VAL(r$)
260 WEND
270 PRINT r$:coup=r:jo$="hum"
280 GOSUB 760:' Jeu
290 IF bol(captj)>24 OR bol(captor)>24 O
R (bol(captj)=24 AND bol(captor)=24)THEN
 440
300 '
310 ' Ordinateur
320 '
330 som=0:FOR i=ns+1 TO 2*ns:som=som+bol
(i):NEXT
340 IF som=0 THEN FOR i=1 TO ns:bol(capt
or)=bol(captor)+bol(i):bol(i)=0:NEXT:GOS
UB 1010:GOTO 440
350 li=21:GOSUB 1390:LOCATE 5,li:PRINT "
J'y pense..."
360 jo$="ord":GOSUB 1170:'Exploration
```

```
370 GOSUB 1390:LOCATE 5,li:PRINT "Je jou
e le coup";coupopt+6
380 coup=coupopt
390 GOSUB 760:' Jeu
400 IF bol(captj)>24 OR bol(captor)>24 O
R (bol(captj)=24 AND bol(captor)=24) THE
N 440 ELSE GOTO 200
410 '
420 ' Fin de jeu
430 '
440 ecart=bol(captor)-bol(captj):li=22:G
OSUB 1390:li=21:GOSUB 1390:LOCATE 5,li
450 IF ecart=0 THEN PRINT "Partie nulle.
" ELSE IF ecart>0 THEN GOSUB 1440 ELSE G
OSUB 1510
460 LOCATE 5,22:PRINT "Voulez vous rejou
er ?":r$=""
470 WHILE r$<>"n" AND r$<>"o"
480   r$=INKEY$:r$=LOWER$(r$)
490 WEND
500 IF r$="o" THEN CLEAR:GOTO 10
510 END
520 '
530 '        CADRE DU JEU
540 '
550 LOCATE 14,5:PRINT "AMSTRAD"
560 LOCATE 30,6:PRINT "CAPTURES"
570 LOCATE 28,10:PRINT "AMSTRAD:":LOCATE
 31,13:PRINT "VOUS:"
580 LOCATE 16,19:PRINT "VOUS"
590 ORIGIN x0,y0
600 FOR y=0 TO 2
610   PLOT 0,y*del:DRAWR 288,0
620 NEXT y
630 FOR x=0 TO 6
640   PLOT x*del,0:DRAWR 0,96
650 NEXT x
660 PAPER 1:PEN 0
670 FOR x=0 TO 5
680   LOCATE 9+x*3,7:PRINT 12-x
690   LOCATE 9+x*3,17:PRINT 1+x
700 NEXT x
710 PAPER 0:PEN 1
720 RETURN
730 '
740 '        JEU
750 '
760 FOR i=1 TO nbol:cf(i)=bol(i):NEXT
770 GOSUB 850:' Traitement d'un coup
```

```
780 FOR i=1 TO nbol:bol(i)=cf(i):NEXT
790 bol(captj)=bol(captj)+caph:bol(capto
r)=bol(captor)+capo
800 GOSUB 1010:' Affichage des positions
810 RETURN
820 '
830 '        TRAITEMENT D'UN COUP
840 '
850 IF jo$="ord" THEN coup=coup+6
860 nc=cf(coup):cf(coup)=0:y=coup
870 FOR x=1 TO nc
880   y=y MOD 12+1:IF y=coup THEN 880
890   cf(y)=cf(y)+1
900 NEXT
910 caph=0:capo=0
920 IF (y-6.5)*(coup-6.5)>0 THEN RETURN
930 WHILE (y-6.5)*(coup-6.5)<0 AND (cf(y
)=2 OR cf(y)=3)
940   IF coup<=6 THEN caph=caph+cf(y) EL
SE capo=capo+cf(y)
950   cf(y)=0:y=y-1:IF y=0 THEN y=12
960 WEND
970 RETURN
980 '
990 '        AFFICHAGE DES POSITIONS
1000 '
1010 TAG
1020 ORIGIN x0+0.2*del,y0+0.65*del
1030 FOR i=1 TO captor
1040   IF bol(i)=0 THEN bol$(i)="  " ELS
E bol$(i)=STR$(bol(i)):bol$(i)=MID$(bol$
(i),2):IF LEN(bol$(i))=1 THEN bol$(i)="
"+bol$(i)
1050 NEXT i
1060 FOR i=1 TO ns
1070   MOVE (i-1)*del,0:PRINT bol$(i);
1080   MOVE (i-1)*del,del:PRINT bol$(13-
i);
1090 NEXT i
1100 TAGOFF
1110 LOCATE 36,10:PRINT bol$(captor)
1120 LOCATE 36,13:PRINT bol$(captj)
1130 RETURN
1140 '
1150 '        EXPLORATION
1160 '
1170 ev0=-30000
1180 FOR c=1 TO ns
1190   FOR i=1 TO nbol:cf(i)=bol(i):NEXT
```

```
1200   IF cf(c+6)=0 THEN 1230
1210   coup=c:GOSUB 850:GOSUB 1280
1220   IF ev1>ev0 THEN ev0=ev1:coupopt=c
1230 NEXT
1240 RETURN
1250 '
1260 '        EVALUATION
1270 '
1280 somjh=0:FOR j=1 TO ns:somjh=somjh+c
f(j):NEXT
1290 som=0:sup=0
1300 FOR j=ns+1 TO 2*ns
1310   sup=MAX(sup,cf(j)):som=som+cf(j)
1320 NEXT
1330 ev1=capo+e1*sup+e2*som
1340 IF somjh=0 THEN ev1=ev1-som
1350 RETURN
1360 '
1370 '        EFFACEMENT D'UNE LIGNE
1380 '
1390 LOCATE 5,li:PRINT STRING$(35," ")
1400 RETURN
1410 '
1420 '        L'AMSTRAD GAGNE
1430 '
1440 IF ecart>=10 THEN PRINT "Il faut tr
availler.":RETURN
1450 IF ecart<10 AND ecart>=5 THEN PRINT
 "Sur la bonne voie.":RETURN
1460 PRINT "J'ai gagne mais belle defens
e."
1470 RETURN
1480 '
1490 '        VOUS GAGNEZ
1500 '
1510 IF ABS(ecart)>=10 THEN PRINT "Since
res felicitations.":RETURN
1520 IF ABS(ecart)<10 AND ABS(ecart)>=5
THEN PRINT "Compliments mais je reviendr
ai.":RETURN
1530 PRINT "Dur combat mais vous etes me
illeur."
1540 RETURN
```

L'AMSTRAD S'ENVOLE AVEC AWELE2

Si vous êtes novice, AWELE1 est exactement ce qu'il vous faut. Ce partenaire modeste mais infatigable accompagnera vos débuts sans créer frustrations ni complexes. De plus, il ne vous fera pas attendre. Au bout de quelques heures, cependant, vous voudrez sans doute viser plus haut. Vous commencerez peut-être (ce serait une excellente idée) par expérimenter de nouvelles formes de la fonction d'évaluation ou de nouveaux paramètres de celle-ci à partir d'idées glanées dans les pages précédentes ou des vôtres propres.

Au-delà, essayez AWELE2. Il diffère de son aîné par deux sous-programmes seulement, dont voici les nouvelles versions.

Variables complémentaires

CF1(NBOL), CF2(NBOL)	: tables mémorisant les configurations aux divers niveaux de l'arbre de jeu.
CAP1, CAP2, CAP3	: captures simulées sur trois coups successifs.
EV1, EV2, EV3	: évaluations aux trois niveaux.
J1, J2, J3	: indices de boucles.

```
1140 '
1150 '          EXPLORATION
1160 '
1170 ev0=-30000
1180 FOR j1=1 TO ns
1190   ev1=30000
1200   FOR i=1 TO nbol:cf(i)=bol(i):NEXT
:IF cf(j1+6)=0 THEN 1370
1210   jo$="ord":coup=j1:GOSUB 850:IF cf
(1)=0 AND cf(2)=0 AND cf(3)=0 AND cf(4)=
0 AND cf(5)=0 AND cf(6)=0 THEN ev1=-(cf(
7)+cf(8)+cf(9)+cf(10)+cf(11)+cf(12))
1220   cap1=capo:FOR i=1 TO nbol:cf1(i)=
cf(i):NEXT
1230   FOR j2=1 TO ns
1240     ev2=-30000
1250     FOR i=1 TO nbol:cf(i)=cf1(i):NE
XT:IF cf(j2)=0 THEN 1350
1260     jo$="hum":coup=j2:GOSUB 850:cap
2=caph:FOR i=1 TO nbol:cf2(i)=cf(i):NEXT
1270     FOR j3=1 TO ns
1280       FOR i=1 TO nbol:cf(i)=cf2(i):
NEXT:IF cf(j3+6)=0 THEN 1320
```

```
1290          jo$="ord":coup=j3:GOSUB 850:c
ap3=capo:GOSUB 1420
1300          IF ev3>=ev1 THEN 1350
1310          ev2=MAX(ev2,ev3)
1320       NEXT j3
1330       IF ev2<ev0 THEN 1370
1340       ev1=MIN(ev1,ev2)
1350    NEXT j2
1360    IF ev1>=ev0 THEN ev0=ev1:coupopt=
j1
1370 NEXT j1
1380 RETURN
1390 '
1400 '          EVALUATION
1410 '
1420 somjh=0:FOR j=1 TO ns:somjh=somjh+c
f(j):NEXT
1430 som=0:sup=0
1440 FOR j=ns+1 TO 2*ns
1450    sup=MAX(sup,cf(j)):som=som+cf(j)
1460 NEXT
1470 ev3=cap1-cap2+cap3+e1*sup+e2*som
1480 IF somjh=0 THEN ev3=ev3-som
1490 RETURN
```

Le sous-programme EXPLORATION applique très directement, sur 3 niveaux de jeu (ordinateur, joueur humain, ordinateur), les méthodes développées au chapitre 5 et notamment l'algorithme Alpha-beta. Nous rappelons ci-dessous les points essentiels, mais une rapide revue du chapitre 5 serait peut-être de saison.

A chaque niveau sont mémorisés :

- une position en cours ; elle permettra de recalculer, chaque fois que nécessaire, toutes les configurations filles qui en proviennent ;
- une évaluation provisoire ;
- le nombre de captures éventuelles réalisées dans le coup qui a conduit à ce niveau.

Trois boucles, de même structure, sont imbriquées. Pour chaque niveau N et au cours de chaque tour de boucle, sont réalisées les opérations suivantes :

- initialisation de l'évaluation notée EV(N) ;
- rappel de la configuration CF(N−1) mémorisée au niveau précédent ;
- jeu d'un coup par appel du sous-programme TRAITEMENT D'UN COUP, lequel renvoie configuration fille et nombre de captures ;
- mémorisation de ces sorties dans CF(N) et CAP(N).

Au troisième niveau intervient l'évaluation. La fonction correspondante est celle d'AWELE1, à une différence près : elle contient le bilan

des captures sur les trois coups successifs qui ont abouti à la position évaluée.

Les instructions 1300 à 1370 traduisent, comme déjà vu au chapitre 5, la procédure MINIMAX avec élagage.

La dernière instruction de la ligne 1210 mérite une mention spéciale. Elle rappelle la pénalité encourue par le joueur qui ne "donne pas à manger" à l'adversaire et incite fortement l'ordinateur à éviter les situations de ce type (technique des "fenêtres").

Tout est maintenant prêt pour vos fécondes luttes avec l'Amstrad muni d'AWELE2. Le programme "réfléchit" assez vite (10 secondes environ par coup) et devrait être coriace. Par rapport à AWELE1, la qualité du jeu est améliorée de façon spectaculaire. Si, cependant, vous en veniez à bout, ajoutez donc un quatrième niveau. Vous savez, maintenant, comment procéder.

Listing du programme AWELE 2

```
10 DEFINT a-d,f-z:titre$="AWELE2"
20 MODE 1:INK 0,1:INK 1,24:PAPER 0:PEN 1
30 CLS:PRINT:PRINT TAB(20-LEN(titre$)/2)
;titre$:PRINT
40 PLOT 10,10:DRAW 630,10:DRAW 630,398:D
RAW 10,398:DRAW 10,10
50 nbol=12:captj=13:captor=14:ns=6:x0=13
0:y0=170:del=48:e1=0.15:e2=0.08
60 DIM bol(captor),bol$(captor),cf(nbol)
,cf1(nbol),cf2(nbol)
70 GOSUB 550:' Cadre du jeu
80 FOR i=1 TO nbol:bol(i)=4:NEXT
90 bol(captj)=0:bol(captor)=0
100 GOSUB 1010:' Affichage des positions
110 li=21:LOCATE 5,li:PRINT "Voulez vous
 jouer le premier ?";:r$=""
120 WHILE r$<>"n" AND r$<>"o"
130   r$=INKEY$:r$=LOWER$(r$)
140 WEND
150 PRINT r$:GOSUB 1530
160 IF r$="n" THEN 330
170 '
180 ' Joueur humain
190 '
200 som=0:FOR i=1 TO ns:som=som+bol(i):N
EXT
210 IF som=0 THEN FOR i=ns+1 TO 2*ns:bol
(captj)=bol(captj)+bol(i):bol(i)=0:NEXT:
GOSUB 1010:GOTO 440
```

```
220 li=22:GOSUB 1530:' Effacement d'une
ligne
230 LOCATE 5,li:PRINT "Votre choix ?";:r
$="":r=0
240 WHILE r<1 OR r>6 OR bol(r)=0
250   r$=INKEY$:r=VAL(r$)
260 WEND
270 PRINT r$:coup=r:jo$="hum"
280 GOSUB 760:' Jeu
290 IF bol(captj)>24 OR bol(captor)>24 O
R (bol(captj)=24 AND bol(captor)=24)THEN
 440
300 '
310 ' Ordinateur
320 '
330 som=0:FOR i=ns+1 TO 2*ns:som=som+bol
(i):NEXT
340 IF som=0 THEN FOR i=1 TO ns:bol(capt
or)=bol(captor)+bol(i):bol(i)=0:NEXT:GOS
UB 1010:GOTO 440
350 li=21:GOSUB 1530:LOCATE 5,li:PRINT "
J'y pense..."
360 GOSUB 1170:'Exploration
370 GOSUB 1530:LOCATE 5,li:PRINT "Je jou
e le coup";coupopt+6
380 coup=coupopt:jo$="ord"
390 GOSUB 760:' Jeu
400 IF bol(captj)>24 OR bol(captor)>24 O
R (bol(captj)=24 AND bol(captor)=24) THE
N 440 ELSE GOTO 200
410 '
420 ' Fin de jeu
430 '
440 ecart=bol(captor)-bol(captj):li=22:G
OSUB 1530:li=21:GOSUB 1530:LOCATE 5,li
450 IF ecart=0 THEN PRINT "Partie nulle.
" ELSE IF ecart>0 THEN GOSUB 1580 ELSE G
OSUB 1650
460 LOCATE 5,22:PRINT "Voulez vous rejou
er ?":r$=""
470 WHILE r$<>"n" AND r$<>"o"
480   r$=INKEY$:r$=LOWER$(r$)
490 WEND
500 IF r$="o" THEN CLEAR:GOTO 10
510 END
520 '
530 '        CADRE DU JEU
540 '
550 LOCATE 14,5:PRINT "AMSTRAD"
```

```
560 LOCATE 30,6:PRINT "CAPTURES"
570 LOCATE 28,10:PRINT "AMSTRAD:":LOCATE
 31,13:PRINT "VOUS:"
580 LOCATE 16,19:PRINT "VOUS"
590 ORIGIN x0,y0
600 FOR y=0 TO 2
610   PLOT 0,y*del:DRAWR 288,0
620 NEXT y
630 FOR x=0 TO 6
640   PLOT x*del,0:DRAWR 0,96
650 NEXT x
660 PAPER 1:PEN 0
670 FOR x=0 TO 5
680   LOCATE 9+x*3,7:PRINT 12-x
690   LOCATE 9+x*3,17:PRINT 1+x
700 NEXT x
710 PAPER 0:PEN 1
720 RETURN
730 '
740 '        JEU
750 '
760 FOR i=1 TO nbol:cf(i)=bol(i):NEXT
770 GOSUB 850:' Traitement d'un coup
780 FOR i=1 TO nbol:bol(i)=cf(i):NEXT
790 bol(captj)=bol(captj)+caph:bol(capto
r)=bol(captor)+capo
800 GOSUB 1010:' Affichage des positions
810 RETURN
820 '
830 '        TRAITEMENT D'UN COUP
840 '
850 IF jo$="ord" THEN coup=coup+6
860 nc=cf(coup):cf(coup)=0:y=coup
870 FOR x=1 TO nc
880   y=y MOD 12+1:IF y=coup THEN 880
890   cf(y)=cf(y)+1
900 NEXT
910 caph=0:capo=0
920 IF (y-6.5)*(coup-6.5)>0 THEN RETURN
930 WHILE (y-6.5)*(coup-6.5)<0 AND (cf(y
)=2 OR cf(y)=3)
940   IF coup<=6 THEN caph=caph+cf(y) EL
SE capo=capo+cf(y)
950   cf(y)=0:y=y-1:IF y=0 THEN y=12
960 WEND
970 RETURN
980 '
990 '        AFFICHAGE DES POSITIONS
1000 '
```

```
1010 TAG
1020 ORIGIN x0+0.2*del,y0+0.65*del
1030 FOR i=1 TO captor
1040   IF bol(i)=0 THEN bol$(i)="  " ELS
E bol$(i)=STR$(bol(i)):bol$(i)=MID$(bol$
(i),2):IF LEN(bol$(i))=1 THEN bol$(i)="
"+bol$(i)
1050 NEXT i
1060 FOR i=1 TO ns
1070   MOVE (i-1)*del,0:PRINT bol$(i);
1080   MOVE (i-1)*del,del:PRINT bol$(13-
i);
1090 NEXT i
1100 TAGOFF
1110 LOCATE 36,10:PRINT bol$(captor)
1120 LOCATE 36,13:PRINT bol$(captj)
1130 RETURN
1140 '
1150 '        EXPLORATION
1160 '
1170 ev0=-30000
1180 FOR j1=1 TO ns
1190   ev1=30000
1200   FOR i=1 TO nbol:cf(i)=bol(i):NEXT
:IF cf(j1+6)=0 THEN 1370
1210   jo$="ord":coup=j1:GOSUB 850:IF cf
(1)=0 AND cf(2)=0 AND cf(3)=0 AND cf(4)=
0 AND cf(5)=0 AND cf(6)=0 THEN ev1=-(cf(
7)+cf(8)+cf(9)+cf(10)+cf(11)+cf(12))
1220   cap1=capo:FOR i=1 TO nbol:cf1(i)=
cf(i):NEXT
1230   FOR j2=1 TO ns
1240     ev2=-30000
1250     FOR i=1 TO nbol:cf(i)=cf1(i):NE
XT:IF cf(j2)=0 THEN 1350
1260     jo$="hum":coup=j2:GOSUB 850:cap
2=caph:FOR i=1 TO nbol:cf2(i)=cf(i):NEXT
1270     FOR j3=1 TO ns
1280       FOR i=1 TO nbol:cf(i)=cf2(i):
NEXT:IF cf(j3+6)=0 THEN 1320
1290       jo$="ord":coup=j3:GOSUB 850:c
ap3=capo:GOSUB 1420
1300       IF ev3>=ev1 THEN 1350
1310       ev2=MAX(ev2,ev3)
1320     NEXT j3
1330     IF ev2<ev0 THEN 1370
1340     ev1=MIN(ev1,ev2)
1350   NEXT j2
```

```
1360   IF ev1>=ev0 THEN ev0=ev1:coupopt=
j1
1370 NEXT j1
1380 RETURN
1390 '
1400 '        EVALUATION
1410 '
1420 somjh=0:FOR j=1 TO ns:somjh=somjh+c
f(j):NEXT
1430 som=0:sup=0
1440 FOR j=ns+1 TO 2*ns
1450   sup=MAX(sup,cf(j)):som=som+cf(j)
1460 NEXT
1470 ev3=cap1-cap2+cap3+e1*sup+e2*som
1480 IF somjh=0 THEN ev3=ev3-som
1490 RETURN
1500 '
1510 '        EFFACEMENT D'UNE LIGNE
1520 '
1530 LOCATE 5,li:PRINT STRING$(35," ")
1540 RETURN
1550 '
1560 '        L'AMSTRAD GAGNE
1570 '
1580 IF ecart>=10 THEN PRINT "Il faut tr
availler.":RETURN
1590 IF ecart<10 AND ecart>=5 THEN PRINT
 "Sur la bonne voie.":RETURN
1600 PRINT "J'ai gagne mais belle defens
e."
1610 RETURN
1620 '
1630 '        VOUS GAGNEZ
1640 '
1650 IF ABS(ecart)>=10 THEN PRINT "Since
res felicitations.":RETURN
1660 IF ABS(ecart)<10 AND ABS(ecart)>=5
THEN PRINT "Compliments mais je reviendr
ai.":RETURN
1670 PRINT "Dur combat mais vous etes me
illeur."
1680 RETURN
```

Chapitre 7
L'ÉCOLE DES DAMES... ET DE L'AWELE

Rendre un ordinateur capable de tenir tête à une intelligence humaine dans un jeu de réflexion pure est déjà un objectif fascinant. Mais ce serait mieux encore si ladite machine pouvait se perfectionner au cours du jeu et des parties successives, accroître sa compétence, en un mot, apprendre.

Dès les premiers pas de l'Intelligence artificielle, les chercheurs tentèrent d'ouvrir une piste dans cette direction, par le biais des fonctions d'évaluation. Dans un premier temps, l'homme fournit une liste de critères basés, nous l'avons montré, sur la connaissance du jeu et l'expérience des meilleurs joueurs. Reste, et ce point est essentiel, à pondérer ces critères, à déterminer leurs coefficients multiplicateurs au sein de la fonction d'évaluation. Si un programme pouvait, de lui-même, estimer puis améliorer ces coefficients, l'avancée serait substantielle. Le progrès quantitatif, à ce stade, deviendrait vite qualitatif : l'ordinateur, dûment alimenté en critères potentiels, pourrait les trier selon l'ordre d'importance des coefficients, puis conserver, par exemple, les N premiers d'un classement évolutif.

Le premier succès très notable dans cette voie est dû à Samuel et à son programme de checkers (version américaine du jeu de dames) évoqué au début du chapitre 5. Nous en esquisserons ici les traits principaux. Puis, à partir des méthodes générales de la cybernétique (notamment recherche et rétroaction) décrites dans la première partie de cet ouvrage, nous construirons, pour l'awele, une procédure efficace et simple d'apprentissage. Pour vérifier les résultats, nous proposerons enfin à l'Amstrad de jouer contre lui-même, après avoir équipé les deux partenaires artificiels de fonctions d'évaluation différentes.

LE PROGRAMME DE SAMUEL

L'objectif d'un joueur de dames est de gagner, c'est-à-dire d'amener son adversaire dans la situation où il ne pourra plus jouer, par manque de pièces ou impossibilité de les mouvoir. La partie est terminée par l'une des situations suivantes :

- le joueur B est incapable de mouvoir un pion ou une dame : dans ce cas, le joueur A a gagné ;
- cas inverse : A est perdant ;
- l'un et l'autre joueur pourront répéter indéfiniment les mêmes mouvements sans conclure : partie nulle.

La difficulté naît du nombre astronomique des configurations possibles (10^{40} environ), de la faible information des joueurs, en cours de partie, quant à l'influence de chaque coup sur le résultat final et de l'incertitude attachée aux décisions de l'adversaire.

Les méthodes utilisées par Samuel, entre les années 1952 et 1955, se rattachent à trois axes :

a) introduction de critères ou indicateurs intermédiaires utilisables en cours de partie ;

b) simulation des conséquences plusieurs coups à l'avance ;

c) techniques d'apprentissage.

Les deux premiers points nous sont déjà familiers. Le deuxième axe utilisait seulement la technique du Minimax car l'Alpha-beta était encore dans les limbes à cette date.

Indicateurs intermédiaires

La probabilité de gagner du joueur A croît, chacun le sait, avec la différence entre les nombres de pions en jeu pour A et pour B. Un premier critère, pour A, consistera donc à choisir la suite de mouvements qui permettent l'écart maximal des prises en faveur de A.

Mais ce critère est insuffisant. Par exemple, comment choisira-t-on entre des mouvements possibles si aucun d'eux ne permet la prise d'un pion à l'adversaire ? De plus, l'objectif est trop étroit. Considéré seul, il peut conduire à des situations défavorables à moyen terme, bien qu'immédiatement avantageuses.

A travers l'étude des ouvrages professionnels, l'analyse des parties jouées par des champions et les entretiens avec eux, Samuel s'est efforcé de définir un ensemble d'indicateurs, traduits en nombres entiers positifs ou négatifs et propres à définir la valeur stratégique d'une configuration. Il serait d'un intérêt limité de vous infliger la liste complète des 38 critères finalement retenus après nombre de tentatives, d'essais et de modifications. Voici, à titre d'exemple, 5 des plus importants.

Avancement

Cet indicateur est, pour les Noirs, la différence entre le nombre de pions possédés dans les 5e et 6e rangées, d'une part, et le nombre de pions des 3e et 4e rangées, de l'autre (les numéros des rangées croissent dans la direction de marche des pions du joueur).

Blancs

	32		31		30		29
28		27		26		25	
	24		23		22		21
20		19		18		17	
	16		15		14		13
12		11		10		9	
	8		7		6		5
4		3		2		1	

Noirs

Garde arrière

Cet indicateur vaut 1 s'il n'y a pas de dames dans le jeu et si les cases 1 et 3, pour les Noirs (ou 30 et 32 pour les Blancs) sont occupées par des pions.

Contrôle central

L'indicateur vaut (pour les Noirs) autant d'unités qu'il y a de pions de la couleur du joueur dans les cases 11, 12, 15, 16, 20, 21, 24 et 25.

Mobilité

Mesurée par le nombre de cases qui pourraient être immédiatement occupées de façon normale (sans tenir compte des coups de saut) par le camp qui va jouer.

Menace

Mesurée par le nombre de cases qui, après mouvement, pourraient être occupées par un pion capable, au coup suivant, de prendre un pion adverse.

Des quantités, analogues aux précédentes, sont calculées à chaque stade du jeu, pour chaque joueur. Si A doit jouer, l'ordinateur évalue, pour toute configuration C donnée, la différence algébrique D_n des valeurs de l'indicateur I_n calculées respectivement pour A et B. La configuration C est alors valorisée, pour A, par un polynôme de la forme :

$$V(C)=D_0+a_1D_1+a_2D_2+...+a_{16}D_{16}$$

obtenu, à partir de 16 indicateurs parmi les 38 étudiés, en multipliant leurs valeurs par des coefficients positifs ou négatifs $a_1,a_2,...a_{16}$.

Le premier terme D_0 est l'écart, en nombre de pions en jeu, de A par rapport à B, dans la configuration C (les dames étant transformées en pions équivalents selon des formules de détail que nous négligeons). Nous reviendrons tout à l'heure sur le choix des 16 indicateurs et de leurs coefficients.

Si la formule d'évaluation V(C) est définie, chacune des positions, résultant de mouvements légaux des joueurs à partir d'une configuration donnée, peut être valorisée par cette formule.

Simulation des conséquences lointaines

Il est clairement impossible, compte tenu des ordres de grandeur, d'explorer jusqu'au bout toutes les parties réalisables d'un jeu de dames. Cependant, les capacités de mémoire et vitesses de calcul des ordinateurs de l'époque autorisaient l'exploration des conséquences pour un nombre limité de niveaux (une vingtaine environ). A partir d'une configuration C quelconque, une arborescence, comportant tous les choix possibles des joueurs A et B pour les 20 prochains coups, peut être calculée. Elle définit, en particulier, toutes les configurations "partiellement terminales" qui peuvent succéder à C après 20 coups. Celles-ci peuvent être toutes valorisées par la formule V.

L'algorithme du MINIMAX déjà justifié et utilisé (*cf.* chapitres 5 et 6) permet alors de valoriser, par remontée dans l'arbre de jeu, toutes les configurations qui peuvent être atteintes en un coup à partir de C, de déterminer la meilleure et de jouer, en conséquence, le prochain coup.

Procédures d'apprentissage

Le programme de Samuel utilisait deux telles procédures.

La première est simple mais fort gourmande en mémoire. Elle consiste, pour toute position valorisée par la technique MINIMAX, à

mémoriser cette configuration et son évaluation. Une bibliothèque de positions valorisées (par exploration de l'arbre à N niveaux) est ainsi constituée et mise à jour. Chaque fois, au cours des parties ultérieures, qu'une telle position apparaîtra en configuration terminale, elle sera évaluée, non pas grâce au polynôme V, mais par appel à la bibliothèque. Pour la position initiale, tout se passe donc comme si l'horizon exploré avait été doublé. Il s'agit bien d'un apprentissage, d'ailleurs efficace, basé sur la mémoire "brute" et l'expérience acquise. Mais cette méthode n'est pas vraiment adaptée à la taille de mémoire d'un ordinateur individuel (sans mémoire à bulles !).

La deuxième procédure est plus intéressante. 38 indicateurs, nous l'avons vu, ont été empiriquement définis. Comment déterminer, à partir d'eux, une bonne formule d'évaluation, un polynôme V efficace ? Comment, en d'autres termes, retenir dans V les meilleurs termes D_n et déterminer leurs pondérations sous la forme de coefficients a_n ?

Ici apparaît l'idée la plus remarquable du programme construit par Samuel. Elle s'exprime de la manière suivante.

Supposons choisi un polynôme V (types d'indicateurs et coefficients). A toute configuration C nous pouvons, alors, associer *deux* évaluations :

- une, soit V, obtenue par calcul direct des indicateurs attachés à C ;
- l'autre, soit V', déterminée par retour en arrière, à partir des configurations terminales au bout de 20 coups (par exemple) joués à partir de C, configurations valorisées par le polynôme choisi.

Si le polynôme d'évaluation était parfait, ces deux valeurs *coïncideraient*, au terme D_0 près, lié au nombre de pions en jeu, terme qui tend à croître en valeur absolue à mesure que se déroule la partie. La valeur de V' devrait d'ailleurs rester constante pour toutes les configurations qui se succèdent dans une partie jouée de façon optimale. Calculée par une procédure MINIMAX, V' apparaît plus sûre que l'indicateur V.

La différence $\Delta = V' - V$, pour une même configuration C, peut donc être un indice de qualité de la formule d'évaluation. Je dis *peut être.* Nous n'en sommes pas sûrs, puisque le polynôme V est justement imparfait. Il s'agit d'une "heuristique", procédant d'une forme d'intuition (humaine, pour l'instant au moins) et justifiée si elle réussit (ou presque).

Admettons cette hypothèse. Il faut alors rechercher les modifications, d'indicateurs ou de coefficients, *dans le sens qui réduit l'écart* Δ. Si $V' - V$, c'est-à-dire Δ, était positif, Samuel en déduisait que V était trop petit. Les termes de la fonction d'évaluation dont la contribution était positive devaient donc être affectés de coefficients plus forts et les critères à coefficients négatifs de poids moins importants. Conclusions inverses si Δ était négatif. Esquissons, sans entrer dans les détails, la procédure pratique utilisée.

Partons d'une formule définie d'évaluation : polynôme V choisi *a priori* ou résultant de modifications précédentes. Après chaque coup joué, l'ordinateur détermine les valeurs et signes des termes D_n et de l'écart Δ. Pour chaque terme D_n sont calculés, sur la base des coups précédemment joués, une statistique des fréquences des signes + ou − et un coefficient de corrélation c_n entre les signes de D_n et Δ. Une valeur c_n positive signifie que D_n et Δ sont, le plus souvent, positifs ou négatifs en même temps. Supposons D_n positif. Alors, en probabilité, Δ l'est aussi et V doit être augmenté, ce qui peut être réalisé par accroissement de a_n. Si, d'autre part, D_n est négatif, Δ l'est, sans doute, aussi ; V est trop fort et doit être diminué, ce qui peut encore être effectué en augmentant a_n.

Conclusion : les coefficients a_n doivent être du même signe que les c_n et croître ou décroître avec eux. Dans le programme de Samuel, les valeurs des coefficients a_n étaient déterminées à un coefficient multiplicatif près. Soit a_1 celui qui correspondait à la plus forte valeur positive c_1 des coefficients de corrélation. La valeur 2^{18} lui était arbitrairement affectée. Les a_n étaient recalculés, après chaque coup, par la formule :

$$a_n = a_1 * \text{Signe}(c_n/c_1)/2^p$$

où p est la partie entière de la valeur absolue du rapport c_1/c_n.

Par exemple, pour $c_1/c_n = 15{,}8$ on trouve $a_n = 2^3$.

La technique précédente suppose choisie une liste de 16 termes actifs parmi les 38 possibies. Le renouvellement de cette liste s'effectue par rotation : si un terme a occupé, pendant huit coups successifs, la dernière position dans le classement, par valeur absolue décroissante, des coefficients de corrélation, il est rayé de la liste des termes actifs, affecté à la liste de réserve en queue de cette liste, et remplacé par le terme placé en tête de cette liste de réserve.

Point important : l'ordinateur apprend quelque chose à chaque coup joué. Il peut donc accroître assez rapidement son expérience. La partie délicate, pour toute procédure d'apprentissage qui implique un feed-back, est d'atténuer les instabilités ou oscillations qui peuvent apparaître, si les coefficients sont mal ajustés. Dans le cas étudié, ces coefficients sont définis par essais successifs, suivis de corrections. Des variations importantes dans la liste des termes retenus et leurs coefficients interviennent, en effet, pendant les 30 premières parties. Mais ensuite, les termes et coefficients deviennent plus stables et le jeu de la machine apparaît meilleur, surtout dans le corps d'une partie. Les finesses des débuts et fins de jeu sont, par contre, difficilement apprises par la méthode qui vient d'être décrite.

AUTOFORMATION DE L'AMSTRAD

Rappelons notre objectif. Nous fournissons une liste de critères et des estimations initiales, éventuellement arbitraires, des coefficients de la fonction d'évaluation. Nous cherchons une méthode, incarnée dans un programme, telle que l'Amstrad puisse, de lui-même, au cours d'une ou de plusieurs parties, améliorer son jeu en modifiant les coefficients de la fonction d'évaluation.

Lignes directrices

Notre solution sera différente de celle de Samuel. Seul point commun, le point de départ :

a) comparer, pour certaines configurations, deux évaluations, l'une directe V, l'autre V', établie selon l'algorithme MINIMAX ;

b) ajuster les coefficients de la fonction d'évaluation afin d'annuler (ou de réduire le plus possible) l'écart $\Delta = V' - V$.

Il convient, pour un apprentissage rapide, de multiplier les comparaisons de ce type. A cette fin, nous déterminerons un écart, non seulement pour chaque coup joué par l'ordinateur mais pour tous les successeurs (niveau 1) de ces coups tels qu'ils apparaissent lors du développement exploratoire sur 3 niveaux. Ainsi, chaque configuration actuelle (niveau 0) sera, dans le processus d'exploration, suivie de N coups (6 au plus). Pour chacun d'eux (niveau 1) seront calculés :

- l'évaluation directe ;
- l'évaluation "remontée" à partir du niveau 3 et donc établie sur un horizon de 2 coups ;
- l'écart entre ces deux valorisations.

La technique d'ajustement provient, en ligne directe, des méthodes générales de la cybernétique abordées dans la première partie de cet ouvrage et, notamment, de la rétroaction (ou feedback) avec apprentissage. Le chapitre 2 a ainsi montré comment résoudre selon ces techniques (programme ZERO), une équation générale à une inconnue :

$$Y(X)=0$$

par itérations.

A chaque tour de boucle nous disposons de deux points :

(X0, Y0) et (X1, Y1)

Est ensuite calculée la pente P de la droite joignant ces deux points.

$$P=(Y1-Y0)/(X1-X0)$$

Puis les deux valeurs X0 et X1 de la variable X sont ajustées :

```
X0=X1
X1=X1-K*SGN(P)*Y1
```

où SGN est la fonction signe du BASIC et K un coefficient positif de rétroaction.

Cette méthode peut être étendue à plusieurs variables. Dans le contexte actuel et avec les notations du futur programme seront effectuées, pour chaque coup analysé, les deux évaluations :

EV1D : évaluation directe au niveau 1
EV1R : évaluation "remontée" au niveau 1

et leur différence :

```
ER=EV1R-EV1D
```

La fonction d'évaluation contient des critères, en nombre NCRIT, et un nombre égal de coefficients mémorisés dans une table E(NCRIT). L'erreur ER est fonction des variables E() (et d'un certain nombre d'autres paramètres). Il s'agit, en bref, de résoudre l'équation :

```
ER=0
```

par rapport aux inconnues E().

Généralisons la procédure utilisée pour une inconnue. A chaque pas des itérations, l'Amstrad aura mémorisé :

- des couples EA(), E() en nombre NCRIT, où EA() est l'ancienne valeur de E() ;
- un coule ERA,ER. Ici encore, ERA est l'ancienne valeur de ER.

La "corrélation" (valeur approchée d'une dérivée partielle, analogue à notre pente P) relative à chaque variable E d'indice K, sera définie par :

```
ECOR=(ER-ERA)/(E(K)-EA(K))
```

Cela fait, les formules d'itération s'écriront, pour tout indice K, de 1 à NCRIT :

```
EA(K)=E(K)
E(K)=E(K)-SGN(ECOR)*ERETRO*ER*RND
ERA=ER
```

avec les notations :

SGN() : fonction signe du BASIC
ERETRO : coefficient positif de rétroaction, à définir pour un bon compromis entre précision et stabilité.

La fonction RND, générateur de nombres aléatoires entre 0 et 1, est ici introduite pour une meilleure exploration du domaine des variables E(). Faute de quoi, les cheminements s'effectueraient uniquement selon des "diagonales" de l'espace des inconnues. La variable ECOR intervient seulement par son signe. Aussi, pour éviter d'éventuelles divisions par zéro, le quotient qui la définit est-il remplacé par un produit.

Le programme AWELEVOL

Prenant AWELE2 comme point de départ, la mise en œuvre de ces idées modifiera les sous-programmes EXPLORATION et EVALUATION. Elle créera deux modules : EVALUATION DIRECTE et APPRENTISSAGE.

Nous avons, à titre d'essai, ajouté aux deux critères de base SUP et SOM, un troisième, NPR, égal au nombre de cases, côté ordinateur, qui, pour la configuration évaluée, contiennent 1 ou 2 pions. NPR est donc un indice de faiblesse de la position. Cela fait (voir listes ci-dessous), les sous-programmes EVALUATION et EVALUATION DIRECTE calculent, en sus du terme normal EV3, une valeur EV3R qui exclut les captures.

```
1440 '
1450 '          EVALUATION
1460 '
1470 somjh=0:FOR j=1 TO ns:somjh=somjh+c
f(j):NEXT:ev3r=0
1480 som=0:sup=0:npr=0
1490 FOR j=ns+1 TO 2*ns:sup=MAX(sup,cf(j
)):som=som+cf(j):IF cf(j)=1 OR cf(j)=2 T
HEN npr=npr+1
1500 NEXT
1510 crit(1)=sup:crit(2)=som:crit(3)=npr
1520 FOR k=1 TO ncrit
1530   ev3r=ev3r+e(k)*crit(k)
1540 NEXT k
1550 ev3=ev3r+cap1-cap2+cap3
1560 IF somjh=0 THEN ev3=ev3-som
1570 RETURN
```

```
1770 '
1780 '          EVALUATION DIRECTE
1790 '
1800 somjh=0:FOR j=1 TO ns:somjh=somjh+c
f1(j):NEXT:ev1d=0
1810 som=0:sup=0:npr=0
1820 FOR j=ns+1 TO 2*ns:sup=MAX(sup,cf1(
j)):som=som+cf1(j):IF cf1(j)=1 OR cf1(j)
=2 THEN npr=npr+1
1830 NEXT
1840 x(1)=sup:x(2)=som:x(3)=npr
1850 FOR k=1 TO ncrit
1860   ev1d=ev1d+e(k)*x(k)
1870 NEXT k
1880 IF somjh=0 THEN ev1d=ev1d-som
1890 RETURN
```

Le sous-programme EXPLORATION, en plus des fonctions déjà assurées dans AWELE2 :

- appelle les modules d'évaluation, directe au niveau 1 pour EV1D, puis, au niveau 3, pour EV3R ;
- calcule EV1R, à partir d'EV3R (Minimax sur deux niveaux) ;
- transmet, pour action, au module d'apprentissage, les valeurs EV1D et EV1R.

```
1170 '
1180 '          EXPLORATION
1190 '
1200 ev0=-30000
1210 FOR j1=1 TO ns
1220   ev1=30000:ev1r=30000
1230   FOR i=1 TO nbol:cf(i)=bol(i):NEXT
:IF cf(j1+6)=0 THEN 1420
1240   jo$="ord":coup=j1:GOSUB 880:IF cf
(1)=0 AND cf(2)=0 AND cf(3)=0 AND cf(4)=
0 AND cf(5)=0 AND cf(6)=0 THEN drapeau=1
:ev1=-(cf(7)+cf(8)+cf(9)+cf(10)+cf(11)+c
f(12))
1250   cap1=capo:FOR i=1 TO nbol:cf1(i)=
cf(i):NEXT
1260   GOSUB 1800:' Evaluation directe
1270   FOR j2=1 TO ns
1280     ev2=-30000:ev2r=-30000
1290     FOR i=1 TO nbol:cf(i)=cf1(i):NE
XT:IF cf(j2)=0 THEN 1390
1300     jo$="hum":coup=j2:GOSUB 880:cap
2=caph:FOR i=1 TO nbol:cf2(i)=cf(i):NEXT
1310     FOR j3=1 TO ns
1320       FOR i=1 TO nbol:cf(i)=cf2(i):
NEXT:IF cf(j3+6)=0 THEN 1360
1330       jo$="ord":coup=j3:GOSUB 880:c
ap3=capo:GOSUB 1470
1340       IF ev3>=ev1 THEN 1390
1350       ev2=MAX(ev2,ev3):ev2r=MAX(ev2
r,ev3r)
1360     NEXT j3
1370     IF ev2<ev0 THEN 1420
1380     ev1=MIN(ev1,ev2):ev1r=MIN(ev1r,
ev2r):IF ev1r=30000 OR ev1r=-30000 THEN
ev1r=ev1d
1390   NEXT j2
1400   IF drapeau=1 THEN drapeau=0 ELSE
GOSUB 1930:' Apprentissage
1410   IF ev1>=ev0 THEN ev0=ev1:coupopt=
j1
```

```
1420 NEXT j1
1430 RETURN
```

Voici la liste complète du programme AWELEVOL, précédée de celle des variables supplémentaires utilisées.

Liste des variables

Tables :

CRIT(NCRIT) : valeurs des critères dans l'évaluation au niveau 3.

E(NCRIT), EA(NCRIT) : valeurs actuelles et anciennes des coefficients de la fonction d'évaluation.

X(NCRIT) : valeurs des critères dans l'évaluation directe.

Variables simples :

DRAPEAU : cet indicateur prend la valeur 1 lorsque toutes les cases sont vides côté ordinateur ; la procédure d'apprentissage est alors court-circuitée.

ECOR : indice de corrélation entre critères et écarts d'évaluation.

ER : écart entre les deux évaluations.

ERETRO : coefficient de feedback.

EV1D : évaluation directe, au niveau 1.

EV1R, EV2R : évaluations indirectes (par Minimax).

EV3R : évaluation au niveau 3.

K : indice de boucle.

NPR : nombre de cases contenant 1 ou 2 pions dans le camp de l'ordinateur.

NCRIT : nombre de critères.

Listing du programme AWELEVOL

```
10 DEFINT a-d,f-z:titre$="AWELEVOL"
20 MODE 1:INK 0,1:INK 1,24:PAPER 0:PEN 1
30 CLS:PRINT:PRINT TAB(20-LEN(titre$)/2)
;titre$:PRINT
40 PLOT 10,10:DRAW 630,10:DRAW 630,398:D
RAW 10,398:DRAW 10,10
50 nbol=12:captj=13:captor=14:ns=6:x0=13
0:y0=170:del=48
```

```
60 ncrit=3:eretro=0.05
70 DIM bol(captor),bol$(captor),cf(nbol)
,cf1(nbol),cf2(nbol)
80 DIM e(ncrit),ea(ncrit),crit(ncrit),x(
ncrit)
90 e(1)=0.15:e(2)=0.08:e(3)=-0.1:RANDOMI
ZE TIME
100 GOSUB 580:' Cadre du jeu
110 FOR i=1 TO nbol:bol(i)=4:NEXT
120 bol(captj)=0:bol(captor)=0
130 GOSUB 1040:' Affichage des positions
140 li=21:LOCATE 5,li:PRINT "Voulez vous
 jouer le premier ?";:r$=""
150 WHILE r$<>"n" AND r$<>"o"
160   r$=INKEY$:r$=LOWER$(r$)
170 WEND
180 PRINT r$:GOSUB 1610:' Effacement d'u
ne ligne
190 IF r$="n" THEN 360
200 '
210 ' Joueur humain
220 '
230 som=0:FOR i=1 TO ns:som=som+bol(i):N
EXT
240 IF som=0 THEN FOR i=ns+1 TO 2*ns:bol
(captj)=bol(captj)+bol(i):bol(i)=0:NEXT:
GOSUB 1040:GOTO 470
250 li=22:GOSUB 1610:' Effacement d'une
ligne
260 LOCATE 5,li:PRINT "Votre choix ?";:r
$="":r=0
270 WHILE r<1 OR r>6 OR bol(r)=0
280   r$=INKEY$:r=VAL(r$)
290 WEND
300 PRINT r$:coup=r:jo$="hum"
310 GOSUB 790:' Jeu
320 IF bol(captj)>24 OR bol(captor)>24 O
R (bol(captj)=24 AND bol(captor)=24)THEN
 470
330 '
340 ' Ordinateur
350 '
360 som=0:FOR i=ns+1 TO 2*ns:som=som+bol
(i):NEXT
370 IF som=0 THEN FOR i=1 TO ns:bol(capt
or)=bol(captor)+bol(i):bol(i)=0:NEXT:GOS
UB 1040:GOTO 470
380 li=21:GOSUB 1610:LOCATE 5,li:PRINT "
J'y pense..."
```

```
390 GOSUB 1200:'Exploration
400 li=21:GOSUB 1610:LOCATE 5,li:PRINT "
Je joue le coup";coupopt+6
410 coup=coupopt:jo$="ord"
420 GOSUB 790:' Jeu
430 IF bol(captj)>24 OR bol(captor)>24 O
R (bol(captj)=24 AND bol(captor)=24) THE
N 470 ELSE GOTO 230
440 '
450 ' Fin de jeu
460 '
470 ecart=bol(captor)-bol(captj):li=22:G
OSUB 1610:li=21:GOSUB 1610:LOCATE 5,li
480 IF ecart=0 THEN PRINT "Partie nulle.
" ELSE IF ecart>0 THEN GOSUB 1660 ELSE G
OSUB 1730
490 LOCATE 5,22:PRINT "Voulez vous rejou
er ?":r$=""
500 WHILE r$<>"n" AND r$<>"o"
510   r$=INKEY$:r$=LOWER$(r$)
520 WEND
530 IF r$="o" THEN CLEAR:GOTO 10
540 END
550 '
560 '        CADRE DU JEU
570 '
580 LOCATE 14,5:PRINT "AMSTRAD"
590 LOCATE 30,6:PRINT "CAPTURES"
600 LOCATE 28,10:PRINT "AMSTRAD:":LOCATE
 31,13:PRINT "VOUS:"
610 LOCATE 16,19:PRINT "VOUS"
620 ORIGIN x0,y0
630 FOR y=0 TO 2
640   PLOT 0,y*del:DRAWR 288,0
650 NEXT y
660 FOR x=0 TO 6
670   PLOT x*del,0:DRAWR 0,96
680 NEXT x
690 PAPER 1:PEN 0
700 FOR x=0 TO 5
710   LOCATE 9+x*3,7:PRINT 12-x
720   LOCATE 9+x*3,17:PRINT 1+x
730 NEXT x
740 PAPER 0:PEN 1
750 RETURN
760 '
770 '        JEU
780 '
790 FOR i=1 TO nbol:cf(i)=bol(i):NEXT
```

```
800 GOSUB 880:' Traitement d'un coup
810 FOR i=1 TO nbol:bol(i)=cf(i):NEXT
820 bol(captj)=bol(captj)+caph:bol(capto
r)=bol(captor)+capo
830 GOSUB 1040:' Affichage des positions
840 RETURN
850 '
860 '        TRAITEMENT D'UN COUP
870 '
880 IF jo$="ord" THEN coup=coup+6
890 nc=cf(coup):cf(coup)=0:y=coup
900 FOR x=1 TO nc
910   y=y MOD 12+1:IF y=coup THEN 910
920   cf(y)=cf(y)+1
930 NEXT
940 caph=0:capo=0
950 IF (y-6.5)*(coup-6.5)>0 THEN RETURN
960 WHILE (y-6.5)*(coup-6.5)<0 AND (cf(y
)=2 OR cf(y)=3)
970   IF coup<=6 THEN caph=caph+cf(y) EL
SE capo=capo+cf(y)
980   cf(y)=0:y=y-1:IF y=0 THEN y=12
990 WEND
1000 RETURN
1010 '
1020 '        AFFICHAGE DES POSITIONS
1030 '
1040 TAG
1050 ORIGIN x0+0.2*del,y0+0.65*del
1060 FOR i=1 TO captor
1070   IF bol(i)=0 THEN bol$(i)="  " ELS
E bol$(i)=STR$(bol(i)):bol$(i)=MID$(bol$
(i),2):IF LEN(bol$(i))=1 THEN bol$(i)="
"+bol$(i)
1080 NEXT i
1090 FOR i=1 TO ns
1100   MOVE (i-1)*del,0:PRINT bol$(i);
1110   MOVE (i-1)*del,del:PRINT bol$(13-
i);
1120 NEXT i
1130 TAGOFF
1140 LOCATE 36,10:PRINT bol$(captor)
1150 LOCATE 36,13:PRINT bol$(captj)
1160 RETURN
1170 '
1180 '        EXPLORATION
1190 '
1200 ev0=-30000
1210 FOR j1=1 TO ns
```

```
1220   ev1=30000:ev1r=30000
1230   FOR i=1 TO nbol:cf(i)=bol(i):NEXT
:IF cf(j1+6)=0 THEN 1420
1240   jo$="ord":coup=j1:GOSUB 880:IF cf
(1)=0 AND cf(2)=0 AND cf(3)=0 AND cf(4)=
0 AND cf(5)=0 AND cf(6)=0 THEN drapeau=1
:ev1=-(cf(7)+cf(8)+cf(9)+cf(10)+cf(11)+c
f(12))
1250   cap1=capo:FOR i=1 TO nbol:cf1(i)=
cf(i):NEXT
1260   GOSUB 1800:' Evaluation directe
1270   FOR j2=1 TO ns
1280     ev2=-30000:ev2r=-30000
1290     FOR i=1 TO nbol:cf(i)=cf1(i):NE
XT:IF cf(j2)=0 THEN 1390
1300     jo$="hum":coup=j2:GOSUB 880:cap
2=caph:FOR i=1 TO nbol:cf2(i)=cf(i):NEXT
1310     FOR j3=1 TO ns
1320       FOR i=1 TO nbol:cf(i)=cf2(i):
NEXT:IF cf(j3+6)=0 THEN 1360
1330       jo$="ord":coup=j3:GOSUB 880:c
ap3=capo:GOSUB 1470
1340       IF ev3>=ev1 THEN 1390
1350       ev2=MAX(ev2,ev3):ev2r=MAX(ev2
r,ev3r)
1360     NEXT j3
1370     IF ev2<ev0 THEN 1420
1380     ev1=MIN(ev1,ev2):ev1r=MIN(ev1r,
ev2r):IF ev1r=30000 OR ev1r=-30000 THEN
ev1r=ev1d
1390   NEXT j2
1400   IF drapeau=1 THEN drapeau=0 ELSE
GOSUB 1930:' Apprentissage
1410   IF ev1>=ev0 THEN ev0=ev1:coupopt=
j1
1420 NEXT j1
1430 RETURN
1440 '
1450 '        EVALUATION
1460 '
1470 somjh=0:FOR j=1 TO ns:somjh=somjh+c
f(j):NEXT:ev3r=0
1480 som=0:sup=0:npr=0
1490 FOR j=ns+1 TO 2*ns:sup=MAX(sup,cf(j
)):som=som+cf(j):IF cf(j)=1 OR cf(j)=2 T
HEN npr=npr+1
1500 NEXT
1510 crit(1)=sup:crit(2)=som:crit(3)=npr
1520 FOR k=1 TO ncrit
```

```
1530   ev3r=ev3r+e(k)*crit(k)
1540 NEXT k
1550 ev3=ev3r+cap1-cap2+cap3
1560 IF somjh=0 THEN ev3=ev3-som
1570 RETURN
1580 '
1590 '        EFFACEMENT D'UNE LIGNE
1600 '
1610 LOCATE 5,li:PRINT STRING$(35," ")
1620 RETURN
1630 '
1640 '        L'AMSTRAD GAGNE
1650 '
1660 IF ecart>=10 THEN PRINT "Il faut tr
availler.":RETURN
1670 IF ecart<10 AND ecart>=5 THEN PRINT
 "Sur la bonne voie.":RETURN
1680 PRINT "J'ai gagne mais belle defens
e."
1690 RETURN
1700 '
1710 '        VOUS GAGNEZ
1720 '
1730 IF ABS(ecart)>=10 THEN PRINT "Since
res felicitations.":RETURN
1740 IF ABS(ecart)<10 AND ABS(ecart)>=5
THEN PRINT "Compliments mais je reviendr
ai.":RETURN
1750 PRINT "Dur combat mais vous etes me
illeur."
1760 RETURN
1770 '
1780 '        EVALUATION DIRECTE
1790 '
1800 somjh=0:FOR j=1 TO ns:somjh=somjh+c
f1(j):NEXT:ev1d=0
1810 som=0:sup=0:npr=0
1820 FOR j=ns+1 TO 2*ns:sup=MAX(sup,cf1(
j)):som=som+cf1(j):IF cf1(j)=1 OR cf1(j)
=2 THEN npr=npr+1
1830 NEXT
1840 x(1)=sup:x(2)=som:x(3)=npr
1850 FOR k=1 TO ncrit
1860   ev1d=ev1d+e(k)*x(k)
1870 NEXT k
1880 IF somjh=0 THEN ev1d=ev1d-som
1890 RETURN
1900 '
1910 '        APPRENTISSAGE
```

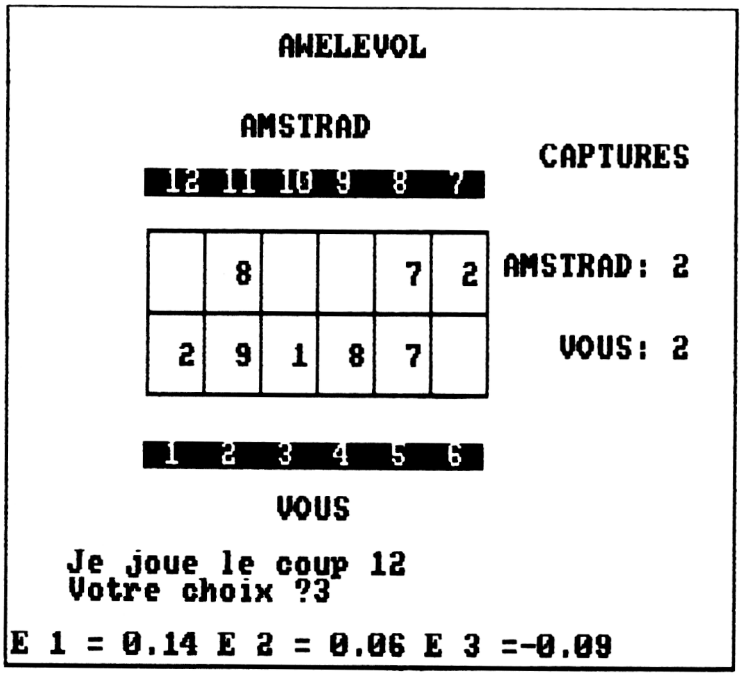
```
1920 '
1930 er=ev1r-ev1d:IF er=0 THEN 1990
1940 FOR k=1 TO ncrit
1950   ecor=(er-era)*(e(k)-ea(k))
1960   ea(k)=e(k):e(k)=e(k)-SGN(ecor)*er
etro*er*RND
1970 NEXT k
1980 era=er
1990 li=24:GOSUB 1610
2000 LOCATE 2,li
2010 FOR k=1 TO ncrit
2020   PRINT "E";k;"=";ROUND(e(k),2);
2030 NEXT k:PRINT
2040 RETURN
```

Voulez-vous jouer avec moi ?

Une fois créé le programme AWELEVOL, vous seriez aimable de jouer avec l'Amstrad afin de jauger ses capacités d'apprentissage et de parfaire son éducation. Le champ d'expérience est vaste. Au premier stade peuvent être initialisés et modifiés, à votre initiative :

– le paramètre ERETRO qui règle l'amplification du feedback ;
– les coefficients E(K) de la fonction d'évaluation.

Si vous lancez alors le programme, les valeurs successives des E(K) s'afficheront plusieurs fois par coup, au bas de l'écran, comme ceci :



Si, par exemple, deux critères, affectés des coefficients E(1) et E(2), ont été choisis, la fonction d'erreur ER, écart entre les évaluations directe et indirecte, peut être représentée par une surface dans l'espace à trois dimensions E(1), E(2), ER. Sauf cas très particulier, la solution de l'équation :

ER= 0

ne sera pas unique. Les solutions se rencontreront sur la courbe d'intersection de la surface représentative et du plan de coordonnées E(1), E(2). Si la surface est très bosselée, et de forme irrégulière, les courbes correspondantes, "lieux géométriques" des solutions, pourront être multiples et non liées entre elles.

Au cours d'une partie, les coefficients E(K) évolueront vers une solution voisine de leurs valeurs initiales, s'il en existe une. Même dans ce cas, les positions finales pourront donc être différentes les unes des autres et dépendront des valeurs initiales. Le point de départ peut également être très éloigné d'une solution, auquel cas le programme se dirigera, d'une démarche hésitante, vers des régions plus favorables, orienté par les lignes de pente de la surface représentative. Sa vitesse de réaction dépendra alors du coefficient ERETRO de feedback. Si ce coefficient est grand (0,5 à 1 par exemple), le programme pourra sauter brusquement d'une région à l'autre ou entrer en oscillations amplifiées (instabilité).

La version présentée du programme met en jeu trois critères liés aux cases sur lesquelles joue l'ordinateur : somme, maximum et cases "faibles". Je vous laisse le plaisir d'explorer à votre gré l'espace des solutions. Voici, à titre de simple information, les résultats de quelques expériences :

- il existe une solution "banale" soit (0, 0, 0), évidemment sans intérêt ;
- partant d'une position (0.2, 0.2, −0.1) définie afin de quitter l'attraction de la zone précédente et pour tenir compte de signes vraisemblables des coefficients, le programme se place et reste dans une zone entourant le point (0.15, 0.08, −0.1) ; c'est une solution probable à confirmer ;
- mais des solutions semblent également exister autour de points tels que les suivants : (0.1, 0.08, −0.11) ou (0.3, 0.22, −0.02) ;
- si l'initialisation part de régions lointaines telles que (1, 1, −0.1), le programme peut parfaitement diverger (oscillations d'amplitudes croissantes) ou tendre vers l'une des valeurs précédentes... ou une autre.

Si vous êtes aventureux, modifiez le coefficient ERETRO, changez le nombre et la nature des critères... et recommencez.

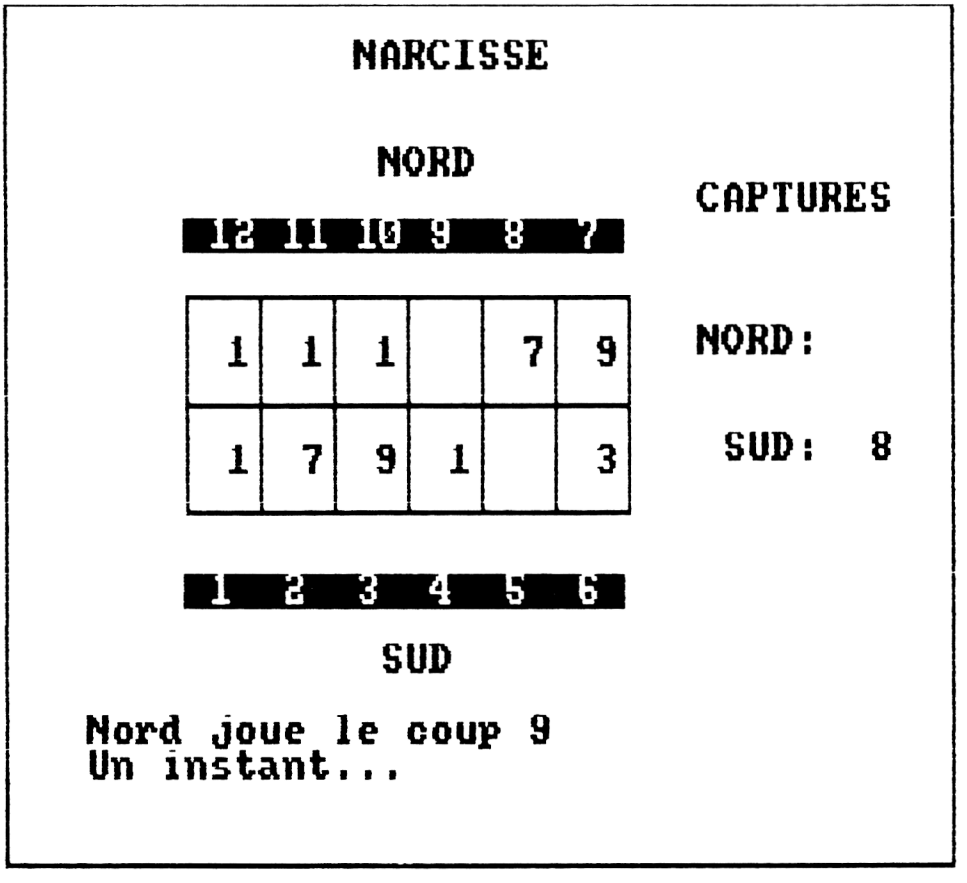
En cas de doute, ou pour départager les ex-aequo, vous pourrez toujours faire jouer le programme contre lui-même avec deux jeux de coefficients. Voici comment.

LES FRÈRES ENNEMIS

La technique d'apprentissage que nous venons d'utiliser est rapide. Elle permet la sélection de zones favorables ou l'élimination de champs stériles. Elle ne saurait cependant remplacer le test final, le combat réel. Il suffit, pour cela, de remplacer le joueur humain, qui tenait la position de SUD, l'ordinateur jouant en NORD, par une autre partition de l'Amstrad. Les modifications correspondantes du programme AWELE2 sont assez simples :

- addition de sous-programmes EXPLORATION SUD et EVALUATION SUD ;
- légers changements dans la présentation et l'architecture générale.

Le résultat, qui n'exige pas de commentaire particulier, est le programme NARCISSE. La copie d'écran et le listing joints le définissent complètement.



La version présentée utilise deux critères : contenu total des cases d'un même camp et valeur maximale, mais il vous sera facile de la modifier pour ajouter des critères ou ajuster les coefficients. Ici NORD joue avec des coefficients nuls (EN1=0, EN2=0). SUD bénéficie, au départ, des valeurs : ES1= 0.15, ES2= 0.08. Le résultat est sans bavure : NORD est, chaque fois, écrasé.

Il serait possible et intéressant de mettre en œuvre, à ce niveau supérieur, une procédure automatique d'apprentissage fondée sur les méthodes déjà utilisées. Pour chaque partie, les adversaires joueraient sur des bases ajustées par le programme lui-même selon les résultats des parties précédentes. Ecrivez ce programme, laissez travailler votre ordinateur, profitez-en pour prendre des vacances et, s'il vous plaît, tenez-moi au courant des résultats.

Listing du programme NARCISSE

```
10 DEFINT a-d,f-z:titre$="NARCISSE"
20 MODE 1:INK 0,1:INK 1,24:PAPER 0:PEN 1
30 CLS:PRINT:PRINT TAB(20-LEN(titre$)/2)
;titre$:PRINT
40 PLOT 10,10:DRAW 630,10:DRAW 630,398:D
RAW 10,398:DRAW 10,10
50 nbol=12:captj=13:captor=14:ns=6:x0=13
0:y0=170:del=48:en1=0:en2=0:es1=0.15:es2
=0.08
60 DIM bol(captor),bol$(captor),cf(nbol)
,cf1(nbol),cf2(nbol)
70 GOSUB 530:' Cadre du jeu
80 FOR i=1 TO nbol:bol(i)=4:NEXT
90 bol(captj)=0:bol(captor)=0
100 GOSUB 990:' Affichage des positions
110 LOCATE 5,21:PRINT "Qui joue le premi
er (n ou s)?";:r$=""
120 WHILE r$<>"n" AND r$<>"s"
130   r$=INKEY$:r$=LOWER$(r$)
140 WEND
150 PRINT r$:li=21:GOSUB 1510:' Effaceme
nt d'une ligne
160 IF r$="n" THEN 310
170 '
180 ' Jeu de SUD
190 '
200 som=0:FOR i=1 TO ns:som=som+bol(i):N
EXT
210 IF som=0 THEN FOR i=ns+1 TO 2*ns:bol
(captj)=bol(captj)+bol(i):bol(i)=0:NEXT:
GOSUB 990:GOTO 420
```

```
220 li=22:GOSUB 1510:LOCATE 5,li:PRINT "
Un instant..."
230 GOSUB 1640:' Exploration SUD
240 GOSUB 1510:LOCATE 5,li:PRINT "Sud jo
ue le coup";coupopt
250 coup=coupopt:jo$="sud"
260 GOSUB 740:' Jeu
270 IF bol(captj)>24 OR bol(captor)>24 O
R (bol(captj)=24 AND bol(captor)=24)THEN
 420
280 '
290 ' Jeu de NORD
300 '
310 som=0:FOR i=ns+1 TO 2*ns:som=som+bol
(i):NEXT
320 IF som=0 THEN FOR i=1 TO ns:bol(capt
or)=bol(captor)+bol(i):bol(i)=0:NEXT:GOS
UB 990:GOTO 420
330 li=21:GOSUB 1510:LOCATE 5,li:PRINT "
Un instant..."
340 GOSUB 1150:'Exploration NORD
350 GOSUB 1510:LOCATE 5,li:PRINT "Nord j
oue le coup";coupopt+6
360 coup=coupopt:jo$="nord"
370 GOSUB 740:' Jeu
380 IF bol(captj)>24 OR bol(captor)>24 O
R (bol(captj)=24 AND bol(captor)=24) THE
N 420 ELSE GOTO 200
390 '
400 ' Fin de jeu
410 '
420 ecart=bol(captor)-bol(captj):li=22:G
OSUB 1510:li=21:GOSUB 1510:LOCATE 5,li
430 IF ecart=0 THEN PRINT "Partie nulle.
" ELSE IF ecart>0 THEN GOSUB 1560 ELSE G
OSUB 1600
440 LOCATE 5,22:PRINT "Un autre essai ?"
:r$=""
450 WHILE r$<>"n" AND r$<>"o"
460   r$=INKEY$:r$=LOWER$(r$)
470 WEND
480 IF r$="o" THEN CLEAR:GOTO 10
490 END
500 '
510 '        CADRE DU JEU
520 '
530 LOCATE 17,5:PRINT "NORD"
540 LOCATE 30,6:PRINT "CAPTURES"
```

```
550 LOCATE 30,10:PRINT "NORD:":LOCATE 31
,13:PRINT "SUD:"
560 LOCATE 17,19:PRINT "SUD"
570 ORIGIN x0,y0
580 FOR y=0 TO 2
590   PLOT 0,y*del:DRAWR 288,0
600 NEXT y
610 FOR x=0 TO 6
620   PLOT x*del,0:DRAWR 0,96
630 NEXT x
640 PAPER 1:PEN 0
650 FOR x=0 TO 5
660   LOCATE 9+x*3,7:PRINT 12-x
670   LOCATE 9+x*3,17:PRINT 1+x
680 NEXT x
690 PAPER 0:PEN 1
700 RETURN
710 '
720 '        JEU
730 '
740 FOR i=1 TO nbol:cf(i)=bol(i):NEXT
750 GOSUB 830:' Traitement d'un coup
760 FOR i=1 TO nbol:bol(i)=cf(i):NEXT
770 bol(captj)=bol(captj)+caph:bol(capto
r)=bol(captor)+capo
780 GOSUB 990:' Affichage des positions
790 RETURN
800 '
810 '        TRAITEMENT D'UN COUP
820 '
830 IF jo$="nord" THEN coup=coup+6
840 nc=cf(coup):cf(coup)=0:y=coup
850 FOR x=1 TO nc
860   y=y MOD 12+1:IF y=coup THEN 860
870   cf(y)=cf(y)+1
880 NEXT
890 caph=0:capo=0
900 IF (y-6.5)*(coup-6.5)>0 THEN RETURN
910 WHILE (y-6.5)*(coup-6.5)<0 AND (cf(y
)=2 OR cf(y)=3)
920   IF coup<=6 THEN caph=caph+cf(y) EL
SE capo=capo+cf(y)
930   cf(y)=0:y=y-1:IF y=0 THEN y=12
940 WEND
950 RETURN
960 '
970 '        AFFICHAGE DES POSITIONS
980 '
990 TAG
```

```
1000 ORIGIN x0+0.2*del,y0+0.65*del
1010 FOR i=1 TO captor
1020   IF bol(i)=0 THEN bol$(i)="  " ELS
E bol$(i)=STR$(bol(i)):bol$(i)=MID$(bol$
(i),2):IF LEN(bol$(i))=1 THEN bol$(i)="
"+bol$(i)
1030 NEXT i
1040 FOR i=1 TO ns
1050   MOVE (i-1)*del,0:PRINT bol$(i);
1060   MOVE (i-1)*del,del:PRINT bol$(13-
i);
1070 NEXT i
1080 TAGOFF
1090 LOCATE 36,10:PRINT bol$(captor)
1100 LOCATE 36,13:PRINT bol$(captj)
1110 RETURN
1120 '
1130 '        EXPLORATION NORD
1140 '
1150 ev0=-30000
1160 FOR j1=1 TO ns
1170   ev1=30000
1180   FOR i=1 TO nbol:cf(i)=bol(i):NEXT
:IF cf(j1+6)=0 THEN 1350
1190   jo$="nord":coup=j1:GOSUB 830:IF c
f(1)=0 AND cf(2)=0 AND cf(3)=0 AND cf(4)
=0 AND cf(5)=0 AND cf(6)=0 THEN ev1=-(cf
(7)+cf(8)+cf(9)+cf(10)+cf(11)+cf(12))
1200   cap1=capo:FOR i=1 TO nbol:cf1(i)=
cf(i):NEXT
1210   FOR j2=1 TO ns
1220     ev2=-30000
1230     FOR i=1 TO nbol:cf(i)=cf1(i):NE
XT:IF cf(j2)=0 THEN 1330
1240     jo$="sud":coup=j2:GOSUB 830:cap
2=caph:FOR i=1 TO nbol:cf2(i)=cf(i):NEXT
1250     FOR j3=1 TO ns
1260       FOR i=1 TO nbol:cf(i)=cf2(i):
NEXT:IF cf(j3+6)=0 THEN 1300
1270       jo$="nord":coup=j3:GOSUB 830:
cap3=capo:GOSUB 1400
1280       IF ev3>=ev1 THEN 1330
1290       ev2=MAX(ev2,ev3)
1300     NEXT j3
1310     IF ev2<ev0 THEN 1350
1320     ev1=MIN(ev1,ev2)
1330   NEXT j2
1340   IF ev1>=ev0 THEN ev0=ev1:coupopt=
j1
```

```
1350 NEXT j1
1360 RETURN
1370 '
1380 '          EVALUATION NORD
1390 '
1400 somjh=0:FOR j=1 TO ns:somjh=somjh+c
f(j):NEXT
1410 som=0:sup=0
1420 FOR j=ns+1 TO 2*ns
1430   sup=MAX(sup,cf(j)):som=som+cf(j)
1440 NEXT
1450 ev3=cap1-cap2+cap3+en1*sup+en2*som
1460 IF somjh=0 THEN ev3=ev3-som
1470 RETURN
1480 '
1490 '          EFFACEMENT D'UNE LIGNE
1500 '
1510 LOCATE 5,li:PRINT STRING$(35," ")
1520 RETURN
1530 '
1540 '          NORD GAGNE
1550 '
1560 PRINT "Nord gagne.":RETURN
1570 '
1580 '          SUD GAGNE
1590 '
1600 PRINT "Sud gagne.":RETURN
1610 '
1620 '          EXPLORATION SUD
1630 '
1640 ev0=-30000
1650 FOR j1=1 TO ns
1660   ev1=30000
1670   FOR i=1 TO nbol:cf(i)=bol(i):NEXT
:IF cf(j1)=0 THEN 1840
1680   jo$="sud":coup=j1:GOSUB 830:IF cf
(7)=0 AND cf(8)=0 AND cf(9)=0 AND cf(10)
=0 AND cf(11)=0 AND cf(12)=0 THEN ev1=-(
cf(1)+cf(2)+cf(3)+cf(4)+cf(5)+cf(6))
1690   cap1=caph:FOR i=1 TO nbol:cf1(i)=
cf(i):NEXT
1700   FOR j2=1 TO ns
1710     ev2=-30000
1720     FOR i=1 TO nbol:cf(i)=cf1(i):NE
XT:IF cf(j2+6)=0 THEN 1820
1730     jo$="nord":coup=j2:GOSUB 830:ca
p2=capo:FOR i=1 TO nbol:cf2(i)=cf(i):NEX
T
1740     FOR j3=1 TO ns
```

```
1750        FOR i=1 TO nbol:cf(i)=cf2(i):
NEXT:IF cf(j3)=0 THEN 1790
1760        jo$="sud":coup=j3:GOSUB 830:c
ap3=caph:GOSUB 1890
1770        IF ev3>=ev1 THEN 1820
1780        ev2=MAX(ev2,ev3)
1790      NEXT j3
1800      IF ev2<ev0 THEN 1840
1810      ev1=MIN(ev1,ev2)
1820    NEXT j2
1830    IF ev1>=ev0 THEN ev0=ev1:coupopt=
j1
1840 NEXT j1
1850 RETURN
1860 '
1870 '        EVALUATION SUD
1880 '
1890 somn=0:FOR j=ns+1 TO 2*ns:somn=somn
+cf(j):NEXT
1900 som=0:sup=0
1910 FOR j=1 TO ns
1920   sup=MAX(sup,cf(j)):som=som+cf(j)
1930 NEXT
1940 ev3=cap1-cap2+cap3+es1*sup+es2*som
1950 IF somn=0 THEN ev3=ev3-som
1960 RETURN
```

TROISIÈME PARTIE
SYSTÈMES EXPERTS

Chapitre 8
CRÉEZ VOS SYSTÈMES EXPERTS PERSONNELS

Les systèmes experts sont à la mode. Dernières créations de l'Intelligence artificielle, ils associent mystère, gloire d'une avancée scientifique et... un zeste de snobisme.

Ces voiles écartés, ils restent des objets fascinants, mais accessibles. Le titre que vous venez de lire n'est point, j'espère vous en convaincre bientôt, illusion ou provocation.

Après les indispensables préliminaires, nous construirons ensemble, dans ce chapitre, trois programmes, de structures voisines, applicables à une grande variété de situations pratiques. Nous en donnerons des exemples.

Le neuvième et dernier chapitre de cet ouvrage vous permettra, ensuite, de créer, également sous forme de programmes, d'autres types de systèmes experts qui acceptent l'incertitude ou sont capables d'apprendre.

PRÉLIMINAIRES

Je crois lire sur vos lèvres diverses questions préalables :

- à quoi servent les systèmes experts ?
- pourquoi et dans quelles conditions vinrent-ils au monde ?
- à quoi ressemblent-ils ?
- quelles en sont les applications principales ?

Tentons quelques réponses.

A quoi servent les systèmes experts ?

A aider les hommes pour la résolution de problèmes qu'ils affrontent dans un domaine précis, technique ou pratique.

Mais cette première réponse est trop vague. Elle s'appliquerait à une calculatrice électronique ou à un programme de facturation. L'aide ou les réponses attendues ne concernent pas, il convient donc de le préciser, les sciences ou technologies mûres telles que mathématiques, physique, navigation astronomique, électronique, résistance des matériaux, etc. Celles-ci s'expriment en méthodes **systématiques et sûres** (en algorithmes) qui utilisent souvent des chiffres et peuvent revêtir la forme de logiciels d'un type classique.

Le champ des systèmes experts est, à l'opposé, celui (qui s'en étonnerait ?), de l'expertise et des experts, celui du savoir-faire, de l'expérience, celui, pour citer quelques exemples, de la médecine, de la programmation, de la prospection minière, du diagnostic de pannes, de l'enseignement assisté, de l'analyse des documents historiques, etc. Dans tous ces domaines, les experts humains sont rares, trop rares. Si, fût-ce dans un domaine très étroit, une fraction de leurs connaissances pouvait être "dupliquée" et mise à la disposition des utilisateurs, les services ainsi rendus pourraient être de grande importance.

De plus (nous le comprendrons mieux dans un instant), un système expert, supposé alimenté en connaissances fragmentées par un expert humain, peut critiquer l'ensemble des règles fournies et déterminer si cet ensemble est, ou non, cohérent. Il dispose, en effet, d'une logique interne qui lui permet de "raisonner" et donc de combiner des prémisses pour en déduire des conclusions.

Les premiers pas

Les pionniers de l'Intelligence artificielle, dans les années 1950, avaient fixé à leur discipline de très ambitieux objectifs : reconnaître, reconstruire les "mécanismes" de la pensée humaine et en simuler les processus. Les résultats spectaculaires et convergents obtenus en quelques années pour les jeux de stratégie (*cf.* deuxième partie de ce livre), la démonstration de théorèmes, etc., avaient donné aux chercheurs de l'époque une vue, sans doute optimiste, des possibilités de la nouvelle science.

Lors de l'extension à des problèmes réels plus vastes et moins bien définis que les jeux ou la logique des propositions, il fallut reconnaître plusieurs difficultés majeures.

La première tient à l'**inculture initiale des ordinateurs.** Un esprit humain ne dispose pas seulement de méthodes intellectuelles. Il amasse au cours du temps une fantastique somme de connaissances qu'il peut appeler, rapprocher, comparer, ordonner, classer, généraliser. Un logiciel ne peut soutenir la comparaison s'il n'est pourvu et ce,

dans un champ nécessairement étroit, d'une somme analogue de savoirs.

Le second obstacle provient des **modes de représentation des connaissances** et des problèmes de communication qu'ils engendrent. Les langues humaines sont par nature (ou nécessité politique et sociale) hautement ambiguës. Leur emploi direct aux fins de communication avec un ordinateur est, pour l'instant, exclu. La traduction automatique, dont les premiers pas datent des années 1950, fut à cet égard un échec exemplaire. Les travaux sur ce sujet, fondés exclusivement sur le vocabulaire et la syntaxe, furent pratiquement abandonnés au cours de la décennie 1960-1970. Il devint évident, à cette époque, qu'une bonne traduction ne pouvait être réalisée sans comprendre le sens du texte, ce qui nous ramène au problème des connaissances.

En troisième lieu, les hommes utilisent, si peu que ce soit, la **logique** naturelle qui permet de déduire des conséquences à partir d'un ensemble d'hypothèses et donc de produire des vérités nouvelles. La seule logique (natale) connue des ordinateurs est la logique binaire. Il convient donc d'établir un pont entre ces deux structures et d'opérer une traduction parfaite de l'une dans l'autre.

Les systèmes experts naquirent précisément de la reconnaissance des obstacles, signalés plus haut, et de tentatives pour les lever.

Structure générale

Le champ d'application est d'abord limité, pour chacun d'eux, à un domaine restreint. Ce sont donc des logiciels très spécifiques. Ils sont ensuite munis d'un faisceau de règles qui condensent le savoir des experts humains dans le type retenu d'activité. C'est **la base de connaissances** ou **base de règles.**

Le traitement efficace des problèmes de représentation et de communication est obtenu par une structuration poussée des concepts relatifs au domaine, des faits évoqués et règles de la base. Chaque proposition aura une forme constante désignant par exemple l'objet en cause, l'aspect sous lequel il est considéré (critère, attribut, caractéristique), la "valeur" prise par ce critère. Les propositions, faits et règles seront traduits en **expressions** dont le vocabulaire et la syntaxe sont parfaitement précisés.

Les systèmes experts sont presque toujours caractérisés par la mise en œuvre d'une logique, sous forme d'un **moteur d'inférence** apte à déduire des conclusions à partir d'un ensemble de faits, qui décrivent une situation donnée, et des règles de la base. Ce module est souvent général et indépendant du champ d'application considéré. Nous en verrons des exemples tout à l'heure. Les logiques utilisées peuvent être binaires (vrai ou faux), admettre l'inconnu ou l'incertain, accepter l'usage de coefficients subjectifs de vraisemblance ou de probabilités (*cf.* chapitre 9). Certains systèmes experts (*cf.* chapitre 9) sont même

capables de construire leurs propres règles de déduction par apprentissage à partir des situations qui leur sont présentées.

Grâce à de telles structures, les systèmes experts opérationnels peuvent être consultés par des utilisateurs humains, dans le domaine de leurs compétences et répondre à l'énoncé d'une situation par un diagnostic, un conseil, une proposition de solution.

La validité de leurs prescriptions repose entièrement, il est à peine besoin de le souligner, sur la qualité du savoir qui leur a été infusé et sur la puissance de leurs mécanismes logiques.

Applications principales

Le plus connu des systèmes experts est MYCIN développé à l'Université de Stanford à partir de 1974. Il traite les infections bactériennes du sang.

Autres systèmes utilisés en médecine :

- INTERNIST (1975) : médecine interne ;
- CASNET (1976) : glaucome de l'œil ;
- PUFF (1978) : infections pulmonaires ;
- ONCOCIN (1981) : cancer ;
- SAM (1981) : hypertension artérielle et maladies vasculaires ;
- SPHINX (1982) : maladies du foie et extensions.

Le plus ancien système expert et l'un des plus remarquables est DENDRAL (1971), construit par Feigenbaum et Buchanan sur les travaux de Lederberg, prix Nobel de médecine. DENDRAL, partant de la formule brute d'une molécule organique et des résultats obtenus au spectrographe de masse, fournit la formule développée de cette molécule.

La prospection pétrolière et la géologie sont également des champs de prédilection pour les systèmes experts. Citons :

- PROSPECTOR (1979) ;
- DIPMETER (1981) ;
- LITHO (1981) ;
- SIMMIAS (1981).

Les autres domaines couverts à ce jour concernent, sans prétention à l'exhaustivité :

- mathématique : élaboration de nouveaux concepts ;
- aide à la conception et à la fabrication mécanique (gammes d'usinage) ;
- électronique ;
- programmation des ordinateurs ;
- diagnostic des pannes d'ordinateur ;
- reconnaissance de la parole ;
- compréhension des langues naturelles ;

– analyse de documents historiques ;
– finance : gestion de portefeuilles boursiers ;
– enseignement assisté ;
– robotique ;
– jeux (échecs, poker, bridge).

VOTRE PREMIER SYSTÈME EXPERT

Eaux vives et langages de programmation

Vous êtes, accordez-le-moi un instant, amateur de navigation et sports nautiques sur fleuves, lacs et rivières. Pour vous renseigner, vous pouvez, naturellement, consulter, l'une après l'autre, les documentations relatives au canal de Bourgogne, au lac du Bourget, aux gorges du Verdon, etc. Mais il serait tellement plus agréable de pouvoir afficher vos souhaits, par exemple :

faire de la planche à voile dans le Sud-Est

et de voir apparaître, dans la seconde suivante, une liste de propositions :

Lac de Roselend
Lac de Serre-Ponçon
Lac de Saint-Cassien
Lac de Sainte-Croix

Nous allons, ensemble, par divers moyens, en commençant par les plus simples, réaliser ce vœu. Mais **le même programme** EXPERS1, et ses différentes versions (EXPERS2, EXPERS3), associés à d'autres "bases de connaissances", peut traiter une infinité de problèmes de cette sorte. Pour vous en convaincre, prenons un second exemple avant de revenir au premier.

Soit les quatre langages de programmation les plus courants : BASIC, PASCAL, FORTRAN, COBOL. Chacun d'eux offre des caractéristiques particulières et se trouvera plus ou moins adapté à certaines applications. Classons ces propriétés selon quatre **critères** :

– *rapidité ;*
– aptitude à la *gestion ;*
– aptitude au *calcul scientifique ;*
– *disponibilité sur 0l* (ordinateur individuel).

Chacun de ces critères est susceptible de prendre un certain nombre de **valeurs** mutuellement exclusives, par exemple :

– *lent, moyen, rapide,* pour le premier critère ;
– *mauvais, moyen, bon,* pour gestion et calcul scientifique ;
– *oui, non,* pour le quatrième critère.

Nous exprimerons sous la forme précédente des **conditions** auxquelles doivent satisfaire les langages que nous pourrons utiliser pour une application définie. Ce sera la "base de faits", traduisant la situation actuelle, le problème à résoudre.

En retour, EXPERS1 fournira des **réponses,** soit la liste du ou des langages informatiques, parmi ceux plus haut cités, qui répondent aux spécifications exprimées.

La "base de connaissances", fournie au programme et que nous appellerons "DATLANG" sera constituée :

- des critères et valeurs ;
- des propriétés associées à chaque langage telles que, par exemple :
  BASIC : lent, moyen en gestion, bon en calcul scientifique, disponible sur OI ;
  Pascal : rapidité moyenne, bon en gestion, moyen en calcul scientifique, disponible sur OI ;
  etc.

Avant d'analyser EXPERS1 et de le construire, voyons ce qu'il nous offre.

Dialogue avec EXPERS1

A l'appel du programme, le message suivant apparaît, complété par notre choix de la base "DATLANG" :

```
                    EXPERS1

 Nommez la base de connaissances que vous souhaitez interroger
parmi celles deja enregistrees sur disquette ou cassette.
 Vous choisirez ensuite,parmi ceux proposes,les criteres et valeurs
qui vous interessent.
 EXPERS1 affichera les reponses qui leur sont associees.

 Nom de la base? datlang

          Travaux preparatoires sur la base

          Quelques secondes SVP.
```

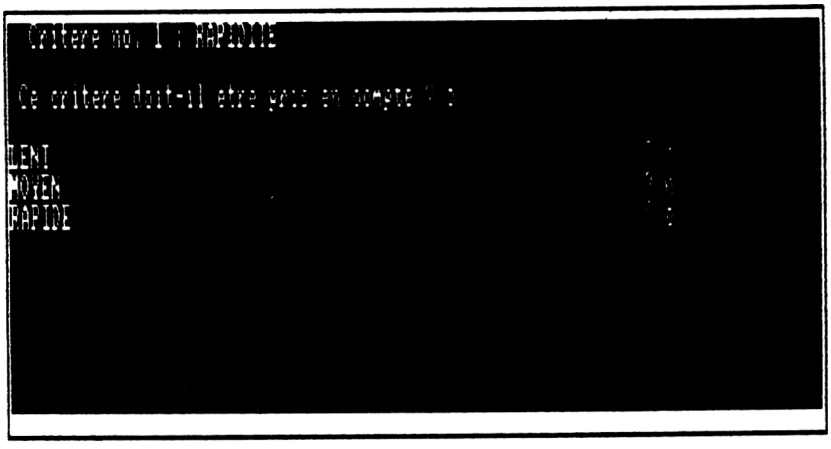
Nous devons ensuite répondre, par oui ou non (o ou n) aux questions posées quant à la pertinence de chaque critère et aux valeurs qui nous intéressent. Si nous voulons seulement un langage rapide et disponible

sur 0l, l'écran offrira, par exemple, les aspects suivants pour les questions, puis les réponses :

SYSTEME EXPERT

Indiquez,pour chaque critere,s'il doit etre pris en compte.
Precisez,pour chaque valeur,si elle doit etre vraie.
Reponses par oui ou non (o ou n).

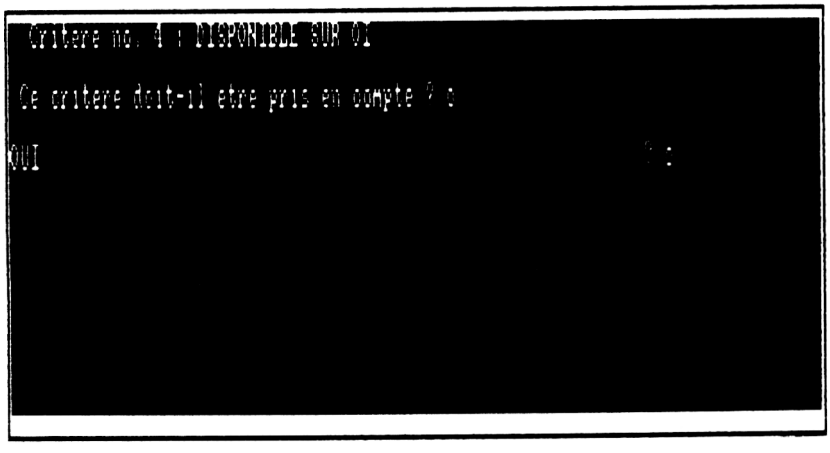
CONSULTATION



SYSTEME EXPERT

Indiquez,pour chaque critere,s'il doit etre pris en compte.
Precisez,pour chaque valeur,si elle doit etre vraie.
Reponses par oui ou non (o ou n).

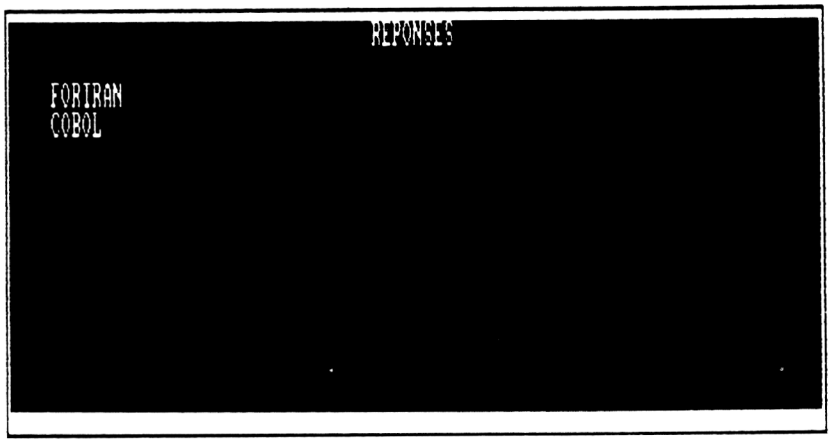
CONSULTATION



SYSTEME EXPERT

Indiquez,pour chaque critere,s'il doit etre pris en compte.
Precisez,pour chaque valeur,si elle doit etre vraie.
Reponses par oui ou non (o ou n).

CONSULTATION



A noter : une valeur choisie pour un critère donné exclut, nous l'avons dit, toutes les autres. La réponse "oui" à une question concernant les valeurs déclenche donc un changement de critère.

Le programme EXPERS1

Dans cette première version de nos systèmes experts, les bases de connaissances utilisées prennent la forme de programmes auxiliaires dont les instructions sont uniquement composées de DATA. Nous en détaillerons dans un instant la forme précise. Le programme lui-même se compose des modules suivants :

a) messages initiaux ; choix, par l'utilisateur, d'une base de connaissances ; entrée, en mémoire centrale, par une instruction CHAIN MERGE, de cette base préalablement enregistrée sur cassette ou disquette.

b) première lecture des DATA afin de déterminer les nombres de critères, de valeurs et de réponses ; dimensions associées des tables d'accueil ;

c) deuxième lecture des DATA ; mémorisation dans les tables ainsi préparées des critères, valeurs et réponses ; codage, sous une forme condensée, des conditions auxquelles satisfont les diverses réponses ;

d) début de la consultation ; entrée, par l'utilisateur, des conditions exigées, en termes de critères et de valeurs ; codage de ces conditions ;

e) balayage des réponses ; sélection et affichage de celles qui répondent aux conditions posées ;

f) retour en *d)* pour une autre consultation ou arrêt du programme au choix de l'utilisateur.

Les bases de connaissances, avons-nous dit, sont condensées en programmes auxiliaires, formés d'instructions DATA. Voici, par exemple, celle, baptisée "DATLANG", relative aux langages de programmation.

```
20000 DATA CRITERES
20010 DATA 1,RAPIDITE
20020 DATA 2,GESTION
20030 DATA 3,CALCUL SCIENTIFIQUE
20040 DATA 4,DISPONIBLE SUR OI
20050 DATA FIN CRITERES
20060 DATA VALEURS
20070 DATA 1,LENT,1
20080 DATA 2,MOYEN,1
20090 DATA 3,RAPIDE,1
20100 DATA 4,MAUVAIS,2
20110 DATA 5,MOYEN,2
20120 DATA 6,BON,2
20130 DATA 7,MAUVAIS,3
20140 DATA 8,MOYEN,3
20150 DATA 9,BON,3
20160 DATA 10,OUI,4
20170 DATA 11,NON,4
20180 DATA FIN VALEURS
20190 DATA REPONSES
20200 DATA BASIC,1,5,9,10,*
20210 DATA PASCAL,2,6,8,10,*
20220 DATA FORTRAN,3,4,9,10,*
20230 DATA COBOL,3,6,7,10,*
20240 DATA FIN REPONSES
```

Les lignes de DATA relatives aux critères sont encadrées par les libellés CRITERES et FIN CRITERES. Chaque ligne est construite selon le format :

DATA n°. d'ordre, libellé

Structure analogue pour les valeurs : encadrement par les libellés VALEURS et FIN VALEURS ; format du modèle suivant :

DATA n°. d'ordre, libellé, n°. du critère concerné

Les réponses, également délimitées par REPONSES et FIN REPONSES, sont construites sur le schéma :

DATA libellé, n°. valeur 1, n°. valeur 2,..., n°. valeur n,*

Les "n° valeur i" sont les numéros d'ordre ou rangs des valeurs exprimant les conditions satisfaites par la réponse où elles interviennent. Par exemple, dans la base "DATLANG", la ligne 20210, relative à Pascal, s'écrit :

```
20210 DATA PASCAL, 2, 6, 8, 10,*
```

où les nombres signifient :

- 2 : rapidité moyenne,
- 6 : bon en gestion,
- 8 : moyen en calcul scientifique,
- 10 : disponible sur 0I.

L'astérisque est un séparateur. Sa rencontre, au cours de la lecture des données, signifiera la fin des informations relatives à la réponse où il intervient.

Une base, telle "DATLANG", une fois créée, sera sauvegardée comme un programme normal, sur cassette ou disquette, de préférence en code ASCII, grâce à la commande SAVE "DATLANG", A. Elle sera appelée et fusionnée avec EXPERS1 par l'instruction 100 qui l'ajoutera au programme :

```
100 CHAIN MERGE A$, 110
```

A$ contient le nom de la base et 110 est le numéro de ligne où se poursuivra l'exécution du programme après fusion.

Voici, construite sur le même modèle, la deuxième base, "DATEAUX", annoncée plus haut.

```
20000 DATA CRITERES
20010 DATA 1,REGION
20020 DATA 2,TYPE D'ACTIVITE
20030 DATA FIN CRITERES
20040 DATA VALEURS
20050 DATA 1,NORD-EST,1
20060 DATA 2,NORD-OUEST,1
20070 DATA 3,SUD-EST,1
20080 DATA 4,SUD-OUEST,1
20090 DATA 5,CENTRE,1
20100 DATA 6,HOUSE-BOAT,2
20110 DATA 7,VOILE,2
20120 DATA 8,PLANCHE A VOILE,2
20130 DATA 9,KAYAK,2
20140 DATA 10,RAFT,2
20150 DATA FIN VALEURS
20160 DATA REPONSES
20170 DATA CANAL DE L'OURCQ,2,6,*
20180 DATA CANAL DU BLAVET,2,6,*
20190 DATA CANAL DE NANTES A BREST,2,6,*
20200 DATA CANAL DE LA LOIRE,5,6,*
```

```
20210 DATA CANAL DU NIVERNAIS,5,6,*
20220 DATA CANAL DE BOURGOGNE,5,6,*
20230 DATA CANAL DU MIDI,4,6,*
20240 DATA CANAL DU RHONE A SETE,3,6,*
20250 DATA LAC DU BOURGET,3,7,*
20260 DATA LAC D'ANNECY,3,7,*
20270 DATA LAC LEMAN,3,7,*
20280 DATA LAC DE ROSELEND,3,8,*
20290 DATA LAC DE SERRE-PONCON,3,8,*
20300 DATA LAC DE VASSIVIERE,5,8,*
20310 DATA LAC DE SAINT-CASSIEN,3,8,*
20320 DATA LAC DE LA FORET D'ORIENT,1,8,
*
20330 DATA LAC DU CHANOINE KIR,1,8,*
20340 DATA LAC DE GEVREY-CHAMBERTIN,5,8,
*
20350 DATA RESERVOIR DE CHAZILLY,5,8,*
20360 DATA ETANG DU DUC,2,8,*
20370 DATA BARRAGE DE VOUGLANS,1,8,*
20380 DATA LAC DE SAINTE-CROIX,3,8,*
20390 DATA ARDECHE (DE VALLON A PONT D'A
RC),3,9,*
20400 DATA VEZERE (PAYSAC),4,9,*
20410 DATA L'OIGNON (DOUBS),1,9,*
20420 DATA GORGES DU TARN,4,9,*
20430 DATA GORGES DU VERDON,3,9,*
20440 DATA UBAYE,3,10,*
20450 DATA HAUT-ALLIER,5,10,*
20460 DATA HAUTE-ISERE,3,10,*
20470 DATA FIN REPONSES
```

Lors de la première lecture des instructions DATA, sont déterminés les nombres N1 de critères, N2 de valeurs et N3 de réponses. En deuxième lecture, les opérations suivantes sont effectuées.

– **Entrée des critères** dans la table CR$(N1) après normalisation de leurs longueurs à L1 (ici 50) caractères, par addition éventuelle de blancs à droite.

– **Mémorisation des valeurs** ; leurs libellés, normalisés à L2 (ici 60) caractères, sont écrits dans la table VL$(N2) ; simultanément le numéro d'ordre de chaque valeur est cadré à droite dans une chaîne de caractères de longueur fixe L3 (ici 3 caractères) ; puis celle-ci est accollée à l'extrémité droite de la chaîne de caractères CR$(I), contenant le critère qui domine chaque valeur ; enfin, le nombre de valeurs dépendant de chaque critère d'indice I est mis à jour dans une table NV(N1) ; ces préparatifs visent à faciliter la présentation ordonnée des critères et valeurs et l'enregistrement des réponses (par oui ou non) de l'utilisateur quant à la pertinence des critères et à la vérité des valeurs.

– **Entrée des réponses** ; leurs libellés sont lus dans la table RS$(N3) ; reste à mémoriser les conditions, définies par les numéros d'ordre des valeurs équivalentes, auxquelles satisfait chaque réponse ; *la représentation utilisée est celle d'un nombre en numération binaire ;* les numéros d'ordre des valeurs définissent les rangs, comptés de droite à gauche, auxquels se placent des chiffres binaires, égaux à 1 si la condition est vérifiée (valeur vraie) et à 0 si elle ne l'est pas ; par exemple, la représentation pour la base "DATLANG", suivant ce code, du langage BASIC, sera :

1100010001

puisque les valeurs de rang 1, 5, 9 et 10 sont vraies pour ce langage ; il est équivalent et plus efficace de traduire en base 10 le nombre ainsi obtenu en numération binaire (soit 785) et de le construire directement par addition des puissances de 2 ; le module "Lecture des réponses", listé ci-dessous, effectue ces opérations et range les nombres décimaux obtenus, pour chaque réponse d'indice K, dans une table RS(N3).

```
560 '
570 '------ Lecture des reponses
580 '
590 READ z$
600 FOR k=1 TO n3
610   READ rs$(k):x=0
620   READ z$:IF z$<>"*" THEN j=VAL(z$):
x=x+2^(j-1):GOTO 620
630   rs(k)=x
640 NEXT
650 READ z$
```

Le noyau du processus de consultation est formé des deux sous-programmes : CONSULTATION et VALEURS CHOISIES, dont liste suit.

```
790 '
800 '        CONSULTATION
810 '
820 CLS:PRINT TAB(33)"SYSTEME EXPERT":PR
INT
830 PRINT " Indiquez,pour chaque critere
,s'il doit etre pris en compte."
840 PRINT " Precisez,pour chaque valeur,
si elle doit etre vraie."
850 PRINT " Reponses par oui ou non (o o
u n).":PRINT
860 PRINT TAB(34)"CONSULTATION"
```

```
870 PLOT 5,35:DRAW 635,35:DRAW 635,260:D
RAW 5,260:DRAW 5,35
880 x=0:wi=1
890 FOR i=1 TO n1
900   CLS#1:GOSUB 1010:' Valeurs choisie
s
910 NEXT i
920 CLS#1:PRINT #1, TAB(36)"REPONSES":PR
INT #1
930 FOR k=1 TO n3
940   IF (rs(k) AND x)=x THEN PRINT #1,
TAB(5);rs$(k)
950 NEXT k
960 wi=0
970 RETURN

980 '
990 '        VALEURS CHOISIES
1000 '
1010 PRINT #1,"  Critere no.";i;": ";LEF
T$(cr$(i),11):PRINT #1
1020 qu$=" Ce critere doit-il etre pris
en compte ? ":GOSUB 1240:PRINT #1
1030 IF UPPER$(r$)="N" THEN RETURN
1040 FOR k1=1 TO nv(i)
1050   j=VAL(MID$(cr$(i),11+1+(k1-1)*13,
13))
1060   qu$=v1$(j)+" ? ":GOSUB 1240
1070   IF UPPER$(r$)="O" THEN x=x+2^(j-1
):RETURN
1080 NEXT k1
1090 RETURN
```

Nous utilisons, pour représenter les choix de l'utilisateur (critères et valeurs), au cours d'une consultation, la même technique de construction d'un nombre binaire et de son équivalent décimal. Les deux boucles :

– lignes 890-910 pour les critères
– lignes 1040-1080 pour les valeurs

permettent, à partir des réponses, Oui ou Non, de l'utilisateur, de calculer un nombre X qui condense les choix effectués et donc les conditions posées.

Les réponses RS$(K), d'indice K, enregistrées dans la base de connaissances, seront ensuite toutes balayées en séquence (instructions 930 à 950), afin de déterminer **celles qui satisfont au moins à toutes les demandes formulées,** mais éventuellement aussi à d'autres conditions.

Quel sera, précisément, le critère de sélection ? Toute condition exprimée se traduit par une position binaire égale à 1 dans l'écriture du nombre caractéristique X. Par exemple, la demande d'un langage de programmation rapide et disponible sur 0I sera associée à une valeur :

X= 1000000100 soit 516 en décimal

Les nombres caractéristiques RS(K) des langages sélectionnés devront contenir, dans leur formule binaire, des chiffres 1 à toutes les positions où ils figurent dans X. Fortran et Cobol vérifient cette condition.

RS(3)= 1100001100 pour Fortran (780 en décimal)
RS(4)= 1001100100 pour Cobol (612 en décimal)

Ce n'est pas le cas de BASIC car :

RS(1)= 1100010001 pour BASIC (785 en décimal)

Le BASIC Amstrad contient une instruction AND qui réalise l'opération logique ET entre les bits de même rang de deux nombres entiers écrits en numération binaire. C'est exactement ce qu'il nous faut, sous la forme (instruction 940) :

RS(K) AND X = X

Pour l'exemple précédent, cela donne :

780 AND 516 = 516
612 AND 516 = 516
785 AND 516 = 512

Fortran et Cobol conviennent donc, selon notre base de connaissances, à la différence de BASIC.

Le reste du programme EXPERS1 est consacré à des opérations classiques de présentation, formatage, ouverture de fenêtre, etc., qui n'appellent pas de commentaires particuliers. Voici donc le listing complet du programme précédé d'une liste des variables.

Liste des variables

Tables :

CR$(N1) : libellés des critères.
NV(N1) : nombres de valeurs associées à chaque critère.
RS(N3) : nombres caractéristiques des réponses.
RS$(N3) : libellés des réponses.
VL$(N2) : libellés des valeurs.

Variables simples :

A$: nom de la base de connaissances utilisée.
CH$: variable intermédiaire pour une chaîne.
I, J, K, K1 : indices de boucles.
L : longueur d'une chaîne de caractères.

L1 : longueur normalisée d'un critère.
L2 : longueur normalisée d'une valeur.
L3 : longueur normalisée d'un nombre.
N1 : nombre de critères.
N2 : nombre de valeurs.
N3 : nombre de réponses.
NN : variable intermédiaire pour un nombre.
NN$: chaîne de caractères associée à NN.
QU$: libellé d'une question.
R : nombre représentant la réponse à une question.
R$: réponse à une question.
X : nombre caractéristique associé à un ensemble de valeurs.
WI : drapeau (0 ou 1) pour impression dans l'une ou l'autre des fenêtres.
Z$: variable intermédiaire de travail.

Listing du programme EXPERS1

```
10 INK 0,0:INK 1,25:PAPER 0:PEN 1:MODE 2
20 CLS:PRINT TAB(36)"EXPERS1":PRINT
30 PRINT " Nommez la base de connaissanc
es que vous souhaitez interroger"
40 PRINT "parmi celles deja enregistrees
 sur disquette ou cassette."
50 PRINT " Vous choisirez ensuite,parmi
ceux proposes,les criteres et valeurs"
60 PRINT "qui vous interessent."
70 PRINT " EXPERS1 affichera les reponse
s qui leur sont associees."
80 PRINT:INPUT" Nom de la base";a$
90 IF LEN(a$)>8 THEN PRINT " Nom de fich
ier trop long !":GOTO 80
100 CHAIN MERGE a$,110
110 PRINT:PRINT TAB(15)"Travaux preparat
oires sur la base ";a$
120 PRINT:PRINT TAB(20)"Quelques seconde
s SVP."
130 DEFINT a-z:l1=50:l2=60:l3=3
140 '
150 '------ Initialisation et dimensions
160 '
170 n1=0:READ z$
180 WHILE UPPER$(z$)<>"FIN CRITERES"
190   READ z$:n1=n1+1
```

```
200 WEND
210 IF (n1 MOD 2)<>1 THEN PRINT " Erreur
 dans la base.":END
220 n1=(n1-1)/2
230 n2=0:READ z$
240 WHILE UPPER$(z$)<>"FIN VALEURS"
250   READ z$:n2=n2+1
260 WEND
270 IF(n2 MOD 3)<>1 THEN PRINT " Erreur
dans la base.":END
280 n2=(n2-1)/3
290 IF n2>15 THEN PRINT " Trop de valeur
s.":END
300 n3=0:READ z$
310 WHILE UPPER$(z$)<>"FIN REPONSES"
320   READ z$:IF z$="*" THEN n3=n3+1
330 WEND
340 DIM cr$(n1),nv(n1),vl$(n2),rs$(n3),r
s(n3)
350 RESTORE
360 '
370 '------ Lecture des criteres
380 '
390 READ z$
400 FOR i=1 TO n1
410   READ i:READ ch$:l=11:GOSUB 1130:'
Longueur normalisee d'une chaine
420   cr$(i)=ch$
430 NEXT
440 READ z$
450 '
460 '------ Lecture des valeurs
470 '
480 READ z$
490 FOR j=1 TO n2
500   READ nn:l=13:GOSUB 1180:' Normalis
ation d'un nombre
510   READ ch$:l=12:GOSUB 1130:' Longueu
r normalisee d'une chaine
520   vl$(j)=ch$
530   READ i:cr$(i)=cr$(i)+nn$:nv(i)=nv(
i)+1
540 NEXT
550 READ z$
560 '
570 '------ Lecture des reponses
580 '
590 READ z$
600 FOR k=1 TO n3
```

```
610   READ rs$(k):x=0
620   READ z$:IF z$<>"*" THEN j=VAL(z$):
x=x+2^(j-1):GOTO 620
630   rs(k)=x
640 NEXT
650 READ z$
660 '
670 ' ------ Appel du systeme expert
680 '
690 WINDOW #1,2,79,10,22:PAPER #1,1:PEN
#1,0
700 GOSUB 820:' Consultation
710 LOCATE 1,24
720 PRINT:qu$=" On continue ? ":GOSUB 12
40:' Reponse par oui ou non
730 IF UPPER$(r$)="N" THEN PRINT " A vot
re service !":END ELSE 700
740 '
750 '
760 '        SOUS-PROGRAMMES
770 '
780 '
790 '
800 '        CONSULTATION
810 '
820 CLS:PRINT TAB(33)"SYSTEME EXPERT":PR
INT
830 PRINT " Indiquez,pour chaque critere
,s'il doit etre pris en compte."
840 PRINT " Precisez,pour chaque valeur,
si elle doit etre vraie."
850 PRINT " Reponses par oui ou non (o o
u n).":PRINT
860 PRINT TAB(34)"CONSULTATION"
870 PLOT 5,35:DRAW 635,35:DRAW 635,260:D
RAW 5,260:DRAW 5,35
880 x=0:wi=1
890 FOR i=1 TO n1
900   CLS#1:GOSUB 1010:' Valeurs choisie
s
910 NEXT i
920 CLS#1:PRINT #1, TAB(36)"REPONSES":PR
INT #1
930 FOR k=1 TO n3
940   IF (rs(k) AND x)=x THEN PRINT #1,
TAB(5);rs$(k)
950 NEXT k
960 wi=0
970 RETURN
```

```
980 '
990 '        VALEURS CHOISIES
1000 '
1010 PRINT #1,"  Critere no.";i;": ";LEF
T$(cr$(i),l1):PRINT #1
1020 qu$=" Ce critere doit-il etre pris
en compte ? ":GOSUB 1240:PRINT #1
1030 IF UPPER$(r$)="N" THEN RETURN
1040 FOR k1=1 TO nv(i)
1050   j=VAL(MID$(cr$(i),l1+1+(k1-1)*13,
13))
1060   qu$=vl$(j)+" ? ":GOSUB 1240
1070   IF UPPER$(r$)="O" THEN x=x+2^(j-1
):RETURN
1080 NEXT k1
1090 RETURN
1100 '
1110 '        LONGUEUR NORMALISEE A L CAR
. D'UNE CHAINE CH$
1120 '
1130 ch$=LEFT$(ch$+STRING$(l,CHR$(32)),l
)
1140 RETURN
1150 '
1160 '        LONGUEUR NORMALISEE D'UN NO
MBRE NN A L CAR. ET JUST. A DROITE
1170 '
1180 nn$=STR$(nn):nn$=MID$(nn$,2)
1190 nn$=RIGHT$(STRING$(l,CHR$(32))+nn$,
l)
1200 RETURN
1210 '
1220 '        REPONSE A UNE QUESTION QU$
PAR OUI OU NON
1230 '
1240 PRINT #wi, qu$;:r$=""
1250 WHILE UPPER$(r$)<>"O" AND UPPER$(r$
)<>"N"
1260   r$=INKEY$
1270 WEND
1280 PRINT #wi,r$:r=VAL(r$)
1290 RETURN
```

FORMEZ DIRECTEMENT VOS BASES DE CONNAISSANCES AVEC EXPERS2

Vous préférez peut-être construire vos bases de connaissances de manière interactive, dans le cadre d'un programme, au lieu de les bâtir sous forme d'une liste de DATA. Si oui, EXPERS2 vous permettra d'avancer de quelques pas vers cet objectif. Son "menu" comporte, bien sûr, l'appel du système expert, pour consultation, mais aussi la création de bases de connaissances.

Si vous choisissez cette option, EXPERS2 vous demandera d'abord une liste de critères. Celle-ci sera immédiatement utilisée pour entrer les valeurs, comme l'indique la copie d'écran ci-dessous :

```
                                    VALEURS

Entrez les valeurs successives associees a chaque critere.
Changez de critere par 'STOP'.

  CRITERE NO. 1 : Rapidite

    > ? Lent
    > ? Moyen
    > ? Rapide
    > ?
```

Restent les réponses. Pour chacun de celles que vous indiquerez, critères et valeurs seront successivement présentés. Vous devrez répondre (*cf.* copie d'écran ci-dessous) :

- Non pour les valeurs qui ne peuvent être associées à la réponse examinée ;
- Oui dans le cas contraire.

```
 Reponse:BASIC

  Critere no. 3 : Calcul scientifique

 Valeurs (Reponse o ou n):

Mauvais
Moyen
Bon
```

Les diverses valeurs s'excluant à l'intérieur d'un même critère, toute réponse Oui déclenchera le passage au critère suivant.

Au terme de ces opérations, le programme écrira, sur cassette ou disquette, un fichier séquentiel, support d'une base de connaissances baptisée, par exemple "LANGAGES" ou "EAUX" et dont voici la structure :

Nombre de caractères
Nombre de valeurs
Pour chaque caractère :
- libellé + numéros d'ordre des valeurs liées
- nombre de valeurs liées

Pour chaque valeur :
- libellé

Nombre de réponses
Pour chaque réponse :
- libellé
- nombre caractéristique associé.

La procédure de consultation est identique à celle d'EXPERS1. Elle débute par la donnée du nom de fichier (base de connaissances) qui vous intéresse. Il est immédiatement recherché, lu et mis à votre disposition. Je vous suggère, à titre d'exercice, de compléter EXPERS2 afin de permettre additions, modifications, suppressions et impressions pour les fichiers, de ce type, utilisés.

Les délais relatifs à EXPERS2 (variables complémentaires et listing) sont donnés ci-dessous.

Liste des variables complémentaires

N1, N2, N3 sont maintenant les limites supérieures affectées respectivement aux critères, valeurs et réponses. Ils servent à dimensionner les tables. Les valeurs réelles, définies au cours du processus interactif d'entrée des données sont mémorisées dans les variables.

NC : nombre de critères.
NV : nombre de valeurs.
NR : nombre de réponses.

Autres variables :

DR : drapeau pour excès de valeurs (NV > 15).
J1 : indice de boucles.
MI, MA : bornes inférieure et supérieure pour un nombre entré au clavier.
Z : variable intermédiaire de travail.

Listing du programme EXPERS2

```
10 INK 0,0:INK 1,13:PAPER 0:PEN 1:MODE 2
20 DEFINT a-z
30 l1=50:l2=60:l3=3
40 n1=10:' Nombre max. de criteres
50 n2=15:' Nombre max. de valeurs (tous
criteres confondus)
60 n3=100:' Nombre max. de reponses
70 DIM cr$(n1),nv(n1),vl$(n2),rs$(n3),rs
(n3)
80 CLS:PRINT TAB(37)"EXPERS2":PRINT:PRIN
T "  Options proposees:"
90 PRINT TAB(5)"1-Creation d'une base de
 connaissances"
100 PRINT TAB(5)"2-Appel du systeme expe
rt"
110 PRINT TAB(5)"3-Fin des operations
120 PRINT:qu$="Votre choix ? "
130 mi=1:ma=3:GOSUB 1660:' Reponse par u
n chiffre
140 ON r GOSUB 240,840,1070
150 GOTO 80
160 '
170 '
180 '        SOUS-PROGRAMMES
190 '
200 '
210 '
220 '        CONSTRUCTION D'UNE BASE
230 '
240 GOSUB 330:' Criteres
250 GOSUB 450:' Valeurs
260 IF dr=1 THEN PRINT "TROP DE VALEURS
!":FOR z=1 TO 3000:NEXT:GOTO 250
270 GOSUB 600:' Reponses
280 GOSUB 1750:' Creation d'un fichier b
ase de connaissances
290 RETURN
300 '
310 '        CRITERES
320 '
330 CLS:PRINT TAB(36)"CRITERES":PRINT
340 PRINT " Entrez les criteres choisis;
frappez 'STOP' pour finir.":PRINT
350 FOR i=1 TO n1
360   PRINT "  CRITERE NO.";i;": ";:INPU
T ch$
```

```
370   IF UPPER$(ch$)="STOP" THEN nc=i-1:
RETURN
380   l=11:GOSUB 1460:'Longueur normalis
ee d'une chaine
390   cr$(i)=ch$
400 NEXT i
410 RETURN
420 '
430 '        VALEURS
440 '
450 j=0:dr=0
460 FOR i=1 TO nc
470   CLS:PRINT TAB(36)"VALEURS":PRINT
480   PRINT " Entrez les valeurs success
ives associees a chaque critere."
490 PRINT " Changez de critere par 'STOP
'.":PRINT
500   PRINT TAB(5)"CRITERE NO.";i;": ";c
r$(i):PRINT
510   GOSUB 1110:' Entree des valeurs
520   IF dr=1 THEN RETURN
530   nv(i)=j1
540 NEXT i
550 nv=j
560 RETURN
570 '
580 '        REPONSES
590 '
600 CLS:PRINT TAB(36)"REPONSES":PRINT
610 PRINT " Entrez chacune des reponses
possibles puis indiquez"
620 PRINT "pour chaque critere et valeur
 s'ils s'appliquent a la reponse examine
e."
630 PRINT " Repondez par oui ou non (o o
u n). Frappez 'STOP' pour finir."
640 PRINT:qu$="Etes-vous pret ? "
650 GOSUB 1570:' Reponse par oui ou non
660 IF UPPER$(r$)="N" THEN 640
670 k=0
680 k=k+1:x=0
690 CLS
700 INPUT" Reponse examinee";z$
710 IF UPPER$(z$)="STOP" THEN nr=k-1:RET
URN
720 rs$(k)=z$
730 FOR i=1 TO nc
740   CLS:PRINT " Reponse:";rs$(k):PRINT
```

```
750   PRINT "  Critere no."i;": ";LEFT$(
cr$(i),11)
760   PRINT:PRINT " Valeurs (Reponse o o
u n):":PRINT
770   GOSUB 1240:' Liaison valeurs-repon
ses
780 NEXT i
790 rs(k)=x
800 GOTO 680
810 '
820 '        CONSULTATION
830 '
840 WINDOW #1,2,79,10,22:PAPER #1,1:PEN
#1,0
850 CLS:PRINT TAB(33)"SYSTEME EXPERT":PR
INT
860 PRINT TAB(34)"CONSULTATION"
870 PRINT " Indiquez,pour chaque critere
,s'il doit etre pris en compte."
880 PRINT " Precisez,pour chaque valeur,
si elle doit etre vraie."
890 PRINT " Reponses par oui ou non (o o
u n).":PRINT
900 INPUT" Nom de la base";a$
910 GOSUB 1940:' Lecture de fichier
920 PLOT 5,35:DRAW 635,35:DRAW 635,260:D
RAW 5,260:DRAW 5,35
930 x=0:wi=1
940 FOR i=1 TO nc
950   CLS #1:GOSUB 1340:' Valeurs choisi
es
960 NEXT i
970 CLS #1:PRINT #1,TAB(36)"REPONSES":PR
INT #1
980 FOR k=1 TO nr
990   IF (rs(k) AND x)=x THEN PRINT #1,T
AB(5);rs$(k)
1000 NEXT k
1010 wi=0:LOCATE 1,24:PRINT STRING$(70,C
HR$(32)):LOCATE 1,24
1020 qu$=" On continue ? ":GOSUB 1570:'
Reponse par oui ou non
1030 IF UPPER$(r$)="N" THEN PRINT " A vo
tre service !":END
1040 LOCATE 1,24:PRINT STRING$(70,CHR$(3
2)):LOCATE 1,24:qu$=" Avec la meme base
? ":GOSUB 1570
1050 IF UPPER$(r$)="O" THEN 930
1060 RETURN
```

```
1070 END
1080 '
1090 '        ENTREE DES VALEURS
1100 '
1110 j1=0
1120 PRINT TAB(8)"> ";:INPUT ch$
1130 IF UPPER$(ch$)="STOP" THEN RETURN
1140 j=j+1:j1=j1+1
1150 IF j>n2 THEN dr=1:RETURN
1160 l=12:GOSUB 1460:' Longueur normalis
ee d'une chaine
1170 vl$(j)=ch$
1180 nn=j:l=13:GOSUB 1510:' Normalisatio
n d'un nombre
1190 cr$(i)=cr$(i)+z$
1200 GOTO 1120
1210 '
1220 '        LIAISON VALEURS-REPONSES
1230 '
1240 FOR k1=1 TO nv(i)
1250   j=VAL(MID$(cr$(i),l1+1+(k1-1)*13,
13))
1260   qu$=vl$(j)+" ? "
1270   GOSUB 1570:' Reponse par oui ou n
on
1280   IF UPPER$(r$)="O" THEN x=x+2^(j-1
):RETURN
1290 NEXT k1
1300 RETURN
1310 '
1320 '        VALEURS CHOISIES
1330 '
1340 PRINT #1,"  Critere no.";i;": ";LEF
T$(cr$(i),l1):PRINT #1
1350 qu$=" Ce critere doit-il etre pris
en compte ? ":GOSUB 1570:PRINT #1
1360 IF UPPER$(r$)="N" THEN RETURN
1370 FOR k1=1 TO nv(i)
1380   j=VAL(MID$(cr$(i),l1+1+(k1-1)*13,
13))
1390   qu$=vl$(j)+" ? ":GOSUB 1570
1400   IF UPPER$(r$)="O" THEN x=x+2^(j-1
):RETURN
1410 NEXT k1
1420 RETURN
1430 '
1440 '        LONGUEUR NORMALISEE A L CAR
. D'UNE CHAINE CH$
1450 '
```

```
1460 ch$=LEFT$(ch$+STRING$(l,CHR$(32)),l
)
1470 RETURN
1480 '
1490 '        LONGUEUR NORMALISEE D'UN NO
MBRE NN A L CAR. ET JUST. A DROITE
1500 '
1510 z$=STR$(nn):z$=MID$(z$,2)
1520 z$=RIGHT$(STRING$(l,CHR$(32))+z$,l)
1530 RETURN
1540 '
1550 '        REPONSE A UNE QUESTION QU$
PAR OUI OU NON
1560 '
1570 PRINT #wi,qu$;:r$=""
1580 WHILE UPPER$(r$)<>"O" AND UPPER$(r$
)<>"N"
1590   r$=INKEY$
1600 WEND
1610 PRINT #wi,r$:r=VAL(r$)
1620 RETURN
1630 '
1640 '        REPONSE A UNE QUESTION PAR
UN CHIFFRE (ENTRE MI ET MA)
1650 '
1660 PRINT qu$;:r$="":r=0
1670 WHILE r<mi OR r>ma
1680   r$=INKEY$:r=VAL(r$)
1690 WEND
1700 PRINT r$
1710 RETURN
1720 '
1730 '        CREATION DU FICHIER A$
1740 '
1750 CLS:INPUT" Nom du fichier a creer";
a$
1760 IF LEN(a$)>8 THEN PRINT "Nom de fic
hier trop long !":PRINT:GOTO 1750
1770 OPENOUT a$
1780 PRINT #9,nc:PRINT #9,nv
1790 FOR i=1 TO nc
1800   PRINT #9,cr$(i):PRINT #9,nv(i)
1810 NEXT i
1820 FOR i=1 TO nv
1830   PRINT #9,vl$(i)
1840 NEXT i
1850 PRINT #9,nr
1860 FOR i=1 TO nr
1870   PRINT #9,rs$(i):PRINT #9,rs(i)
```

```
1880 NEXT i
1890 CLOSEOUT
1900 RETURN
1910 '
1920 '        LECTURE DU FICHIER A$
1930 '
1940 OPENIN a$
1950 INPUT#9,nc,nv
1960 FOR i=1 TO nc
1970   INPUT#9,cr$(i),nv(i)
1980 NEXT i
1990 FOR i=1 TO nv
2000   INPUT#9,vl$(i)
2010 NEXT i
2020 INPUT#9,nr
2030 FOR i=1 TO nr
2040   INPUT#9,rs$(i),rs(i)
2050 NEXT i
2060 CLOSEIN
2070 RETURN
```

UN VÉRITABLE OUTIL D'INTELLIGENCE ARTIFICIELLE : EXPERS3

Un examen attentif des deux programmes précédents vous a sans doute révélé une limitation : le nombre total N2 des valeurs est borné à 15. Raison ? La méthode utilisée, certes élégante et simple, est basée sur la représentation informatique des nombres entiers et les opérateurs logiques (AND en particulier) qui leur sont liés. Or le BASIC de l'Amstrad, comme beaucoup d'autres, limite cette représentation à deux octets, soit 16 bits. D'où... Pour obtenir un programme pratique et général, cette contrainte doit être levée.

Par ailleurs, la sélection des "bonnes" réponses, (celles qui satisfont à des conditions définies) utilise l'exploration en séquence de **toutes** les réponses incluses dans la base et le test de leurs nombres caractéristiques. Adaptée à des bases de petite dimension, cette technique conduirait, sans doute, à des temps de traitement un peu longs pour d'importants volumes de données. Il serait plus efficace de repérer les seules réponses compatibles avec une combinaison définie de critères et valeurs, réponses vers lesquelles "pointe" cette combinaison.

A ce point de notre voyage aux insolites régions des systèmes experts, serait bienvenue une définition abstraite mais précise des problèmes solubles par nos programmes. En voici les éléments.

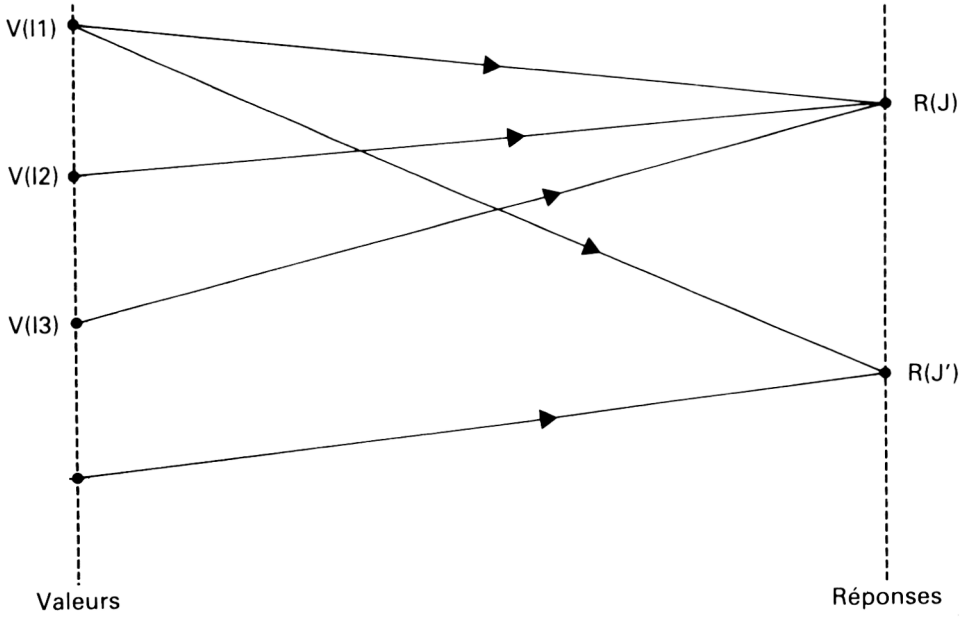
a) Les données de chaque base de connaissances forment **deux groupes distincts et séparés** :

– critères et valeurs,
– réponses.

b) Les règles qui condensent le "savoir" propre au domaine couvert par une base de connaissances sont de la forme :

SI valeur n°. I1 est vraie
ET SI valeur n°. I2 est vraie
ET SI valeur n°. I3 est vraie
ALORS la réponse n°. J est correcte.

Cette structure logique peut être représentée par un "graphe" formé de sommets (ou nœuds) et d'arcs orientés qui les relient. Il ressemblerait, par exemple, à ceci :



A gauche, les sommets V(I) représentatifs des valeurs ; à droite, ceux, R(J), liés aux réponses. La règle logique, formulée plus haut, se traduit par un faisceau d'arcs orientés reliant les sommets V(I1), V(I2), V(I3) à la réponse R(J). Il en serait de même pour une autre règle de forme analogue, telle que :

SI valeur n°.I1 ET valeur n°. I'2 sont vraies
ALORS réponse n°.J'

La structure logique d'une base de connaissances du type étudié associe, dans le graphe, à chaque "sommet-réponse" R(J) une **liste d'antécédents,** soit, dans notre exemple, V(I1), V(I2), V(I3).

Connaissant les antécédents de tous les "sommets-réponses", il devrait être possible de calculer, pour chaque "sommet-valeur", **la liste de ses successeurs** dans le graphe, c'est-à-dire les réponses vers lesquelles pointent les arcs orientés qui partent de ce "sommet-valeur". Par exemple, sur la figure, V(I1)a, pour successeurs, R(J) et R(J').

La construction des deux groupes de listes :

– liste des antécédents pour chaque réponse,
– liste des successeurs pour chaque valeur,

nous fournira un précieux outil d'exploration du graphe en tous sens et toutes directions. Munis de cet instrument, nous pourrons, nous l'allons montrer tout à l'heure, résoudre les problèmes du type suivant :

Voici les conditions posées (liste de valeurs qui doivent être vérifiées). Quelles sont les réponses qui y satisfont ?

Un nombre considérable de questions pratiques entrent dans le cadre ainsi défini et peuvent être traitées par ces méthodes. Citons quelques exemples que vous compléterez aisément :

- recherches documentaires dans les archives, journaux, bibliothèques, filmothèques, etc., à partir de mots clés ;
- sélection, avec accès rapide, de produits, voyages, vacances, restaurants, magasins, associations... ;
- reconnaissance de types : détermination de bactéries ou virus à partir d'observations ; diagnostic de maladies à partir de symptômes... ;
- classement en différentes espèces de minéraux, végétaux, etc., au vu de leurs caractéristiques ;
- organisation personnelle de vos livres, disques, cassettes audio et vidéo, documents de travail, etc., afin de pouvoir retrouver facilement ceux qui vous seront utiles à un instant donné.

Je vais maintenant vous présenter l'outil EXPERS3, qui offre ces possibilités, en insistant sur les caractères particuliers qui le distinguent de ses prédécesseurs, EXPERS1 et EXPERS2.

Structure des données

Afin de conserver la souplesse maximale pour toutes additions, suppressions et modifications dans les bases de connaissances, celles-ci seront constituées de DATA, selon le format déjà employé pour EXPERS1. Ainsi, par exemple, les bases DATLANG et DATEAUX seront utilisables sans problème avec EXPERS3.

A lecture de ces DATA, critères et valeurs seront, comme précédemment, entrés dans les tables CR$(N1) et VL$(N2). Il en ira différemment pour les réponses. Leurs libellés seront toujours placés dans la table RS$(N3), mais une nouvelle table ANT$(N3), de chaînes de caractères, contiendra une **liste de pointeurs** vers les valeurs qui conditionnent chaque réponse. Chaque liste est formée des numéros d'ordre des valeurs et commence par le nombre des pointeurs qu'elle contient.

Par exemple, à partir de la base DATLANG, la réponse BASIC, première du lot, sera traduite en :

- l'élément de table RS$(1) = "BASIC" ;
- l'élément de table ANT$(1)=" 4 1 5 9 10".

Chaque nombre contenu dans ANT$(1) est cadré à droite dans une sous-chaîne de L3 (ici L3=3) caractères. ANT$(1) s'interprète ainsi :

- 4 valeurs antécédentes ;
- leurs numéros d'ordre sont 1, 5, 9, 10.

Nous devons, réciproquement, associer à toute valeur une liste de pointeurs vers les réponses qui constituent ses successeurs. Par définition, une réponse succède à une valeur si celle-ci figure parmi les antécédents de la réponse. Le sous-programme SUCCESSEURS DES VALEURS, dont la liste suit, calcule donc les éléments de la table SUC$(N2), en nombre égal au nombre des valeurs. Chacun d'eux est construit sur un format identique à celui de ANT$(), à savoir des sous-chaînes de L3 caractères, concaténées. Elles contiennent le nombre des successeurs, puis les indices des "réponses-successeurs".

```
1480 '
1490 '       SUCCESSEURS DES VALEURS
1500 '
1510 l=13
1520 FOR k=1 TO n3
1530   nn=k:GOSUB 1360
1540   na=VAL(LEFT$(ant$(k),13))
1550   FOR k1=1 TO na
1560     j=VAL(MID$(ant$(k),k1*13+1,13))
1570     suc$(j)=suc$(j)+nn$:ns(j)=ns(j)
+1
1580   NEXT k1
1590 NEXT k
1600 FOR j=1 TO n2
1610   nn=ns(j):GOSUB 1360:suc$(j)=nn$+s
uc$(j)
1620 NEXT j
1630 RETURN
```

Consultation

Le choix des conditions posées par réponses Oui ou Non à mesure que défilent critères et valeurs s'effectue, avec EXPERS3, comme pour ses prédécesseurs EXPERS1 ET 2. Mais (*cf.* sous-programme VALEURS CHOISIES ci-dessous) :

- les indices J des valeurs sélectionnées sont mémorisés dans une chaîne VR$ de même structure que ANT$() et SUC$(). En particulier, cette chaîne débute par l'indication du nombre NVR des valeurs retenues ;
- les éléments de rang J d'une table d'entiers BITV(N2) sont commutés de 0 à 1 pour ces valeurs sélectionnées.

```
1160 '
1170 '          VALEURS CHOISIES
1180 '
1190 PRINT #1,"  Critere no.";i;": ";LEF
T$(cr$(i),11):PRINT #1
1200 qu$=" Ce critere doit-il etre pris
en compte ? ":GOSUB 1420:PRINT #1
1210 IF UPPER$(r$)="N" THEN RETURN
1220 FOR k1=1 TO nv(i)
1230   j=VAL(MID$(cr$(i),11+1+(k1-1)*13,
13))
1240   qu$=v1$(j)+" ? ":GOSUB 1420
1250   IF UPPER$(r$)="O" THEN bitv(j)=1:
nn=j:l=13:GOSUB 1360:vr$=vr$+nn$:nvr=nvr
+1:RETURN
1260 NEXT k1
1270 RETURN
```

Une fois exprimés les choix complets de l'utilisateur, commence le processus de calcul des réponses grâce à l'algorithme ci-dessous (lignes 1000 à 1130 du sous-programme CONSULTATION). Étapes principales :

a) balayer la chaîne VR$;

b) pour chaque valeur d'indice X de cette chaîne, appeler la liste de pointeurs SUC$(X) et déterminer le nombre NS des successeurs ;

c) pour chaque "réponse-successeur" Y, appeler la liste de pointeurs ANT$(Y) et déterminer le nombre NA de ses antécédents ;

d) pour chaque antécédent Z, déterminer si le "bit de vérité" qui lui est associé, soit BITV(Z), a la valeur 1 ; si oui, un compteur NVR1 est incrémenté d'une unité ;

e) si le nombre des antécédents vrais d'une même réponse est égal au nombre des conditions posées, cette réponse vérifie toutes ces conditions, elle est donc satisfaisante et prend place comme élément d'une table REPONSES$(N3) ; simultanément, un compteur NREPONSES est incrémenté (*cf.* sous-programme LISTE DES REPONSES ci-dessous) ;

f) une même réponse peut apparaître plusieurs fois comme successeur de différentes valeurs, ces éventuelles redondances sont testées et supprimées par examen de la table REPONSES$() avant chaque écriture dans celle-ci.

```
1000 FOR j=1 TO nvr
1010   x=VAL(MID$(vr$,j*13+1,13))
1020   ns=VAL(LEFT$(suc$(x),13))
1030   FOR k=1 TO ns
1040     nvr1=0
1050     y=VAL(MID$(suc$(x),k*13+1,13))
1060     na=VAL(LEFT$(ant$(y),13))
1070     FOR k1=1 TO na
1080       z=VAL(MID$(ant$(y),k1*13+1,13
))
1090       IF bitv(z)=1 THEN nvr1=nvr1+1
1100     NEXT k1
1110     IF nvr1=nvr THEN GOSUB 1670:' L
iste des reponses
1120   NEXT k
1130 NEXT j
```

```
1640 '
1650 '        LISTE DES REPONSES
1660 '
1670 IF nreponses=0 THEN reponses$(1)=rs
$(y):nreponses=1:RETURN
1680 FOR k2=1 TO nreponses
1690   IF reponses$(k2)=rs$(y) THEN RETU
RN
1700 NEXT k2
1710 nreponses=nreponses+1:reponses$(nre
ponses)=rs$(y)
1720 RETURN
```

Le programme EXPERS3

Voici le listing du programme complet, précédé d'une liste de variables supplémentaires propres à EXPERS3.

Liste des variables supplémentaires

Tables :

ANT$(N3)	: antécédents des réponses dans la base de connaissances.
BITV(N2)	: bits de vérité (0 ou 1) attachés aux valeurs choisies.
NS(N2)	: nombre de successeurs des valeurs.
REPONSES$(N3)	: table des réponses qui satisfont aux conditions.
SUC$(N2)	: successeurs des valeurs.

Variables simples :

K2	: indice de boucle.
NA	: nombre des antécédents pour la réponse examinée.
NREPONSES	: nombre des réponses qui satisfont aux conditions posées.
NS	: nombre des successeurs pour la valeur examinée.
NVR	: nombre de valeurs choisies par l'utilisateur comme devant être vraies.
NVR1	: nombre de conditions vérifiées par la réponse examinée.
VR$	: chaîne de caractères contenant les indices des valeurs choisies par l'utilisateur.
X, Y, Z	: indices courants des tables.

Listing du programme EXPERS3

```
10 INK 0,0:INK 1,10:PAPER 0:PEN 1:MODE 2
20 CLS:PRINT TAB(36)"EXPERS3":PRINT
30 PRINT " Nommez la base de connaissanc
es que vous souhaitez interroger"
40 PRINT "parmi celles deja enregistrees
 sur disquette ou cassette."
50 PRINT " Vous choisirez ensuite,parmi
ceux proposes,les criteres et valeurs"
60 PRINT "qui vous interessent."
70 PRINT " EXPERS3 affichera les reponse
s qui leur sont associees."
80 PRINT:INPUT" Nom de la base";a$
```

```
90 IF LEN(a$)>8 THEN PRINT " Nom de fich
ier trop long !":GOTO 80
100 CHAIN MERGE a$,110
110 PRINT:PRINT TAB(15)"Travaux preparat
oires sur la base ";a$
120 PRINT:PRINT TAB(20)"Quelques seconde
s SVP."
130 DEFINT a-z:l1=50:l2=60:l3=3
140 '
150 '------ Initialisation et dimensions
160 '
170 n1=0:READ z$
180 WHILE UPPER$(z$)<>"FIN CRITERES"
190   READ z$:n1=n1+1
200 WEND
210 IF (n1 MOD 2)<>1 THEN PRINT " Erreur
 dans la base.":END
220 n1=(n1-1)/2
230 n2=0:READ z$
240 WHILE UPPER$(z$)<>"FIN VALEURS"
250   READ z$:n2=n2+1
260 WEND
270 IF(n2 MOD 3)<>1 THEN PRINT " Erreur
dans la base.":END
280 n2=(n2-1)/3
290 n3=0:READ z$
300 WHILE UPPER$(z$)<>"FIN REPONSES"
310   READ z$:IF z$="*" THEN n3=n3+1
320 WEND
330 DIM cr$(n1),nv(n1),vl$(n2),ns(n2),su
c$(n2),bitv(n2),rs$(n3),ant$(n3),reponse
s$(n3)
340 RESTORE
350 '
360 '------ Lecture des criteres
370 '
380 READ z$
390 FOR i=1 TO n1
400   READ i:READ ch$:l=l1:GOSUB 1310:'
Longueur normalisee d'une chaine
410   cr$(i)=ch$
420 NEXT
430 READ z$
440 '
450 '------ Lecture des valeurs
460 '
470 READ z$
480 FOR j=1 TO n2
```

```
490   READ nn:l=13:GOSUB 1360:' Normalis
ation d'un nombre
500   READ ch$:l=12:GOSUB 1310:' Longueu
r normalisee d'une chaine
510   vl$(j)=ch$
520   READ i:cr$(i)=cr$(i)+nn$:nv(i)=nv(
i)+1
530 NEXT
540 READ z$
550 '
560 '------ Lecture des reponses
570 '
580 READ z$:l=13
590 FOR k=1 TO n3
600   READ rs$(k):na=0
610   READ z$:IF z$<>"*" THEN nn=VAL(z$)
:GOSUB 1360:ant$(k)=ant$(k)+nn$:na=na+1:
GOTO 610
620   nn=na:GOSUB 1360:ant$(k)=nn$+ant$(
k)
630 NEXT
640 READ z$
650 GOSUB 1510:' Successeurs des valeurs
660 '
670 ' ------ Appel du systeme expert
680 '
690 WINDOW #1,2,79,10,22:PAPER #1,1:PEN
#1,0
700 GOSUB 850:' Consultation
710 FOR k=1 TO nreponses
720   PRINT #1,TAB(5);reponses$(k)
730 NEXT k
740 LOCATE 1,24
750 PRINT:qu$=" On continue ? ":GOSUB 14
20:' Reponse par oui ou non
760 IF UPPER$(r$)="N" THEN PRINT " A vot
re service !":END ELSE 700
770 '
780 '
790 '        SOUS-PROGRAMMES
800 '
810 '
820 '
830 '        CONSULTATION
840 '
850 CLS:PRINT TAB(33)"SYSTEME EXPERT":PR
INT
860 nvr=0:vr$="":reponses$="":nreponses=
0
```

```
870 FOR j=1 TO n2:bitv(j)=0:NEXT
880 PRINT " Indiquez,pour chaque critere
,s'il doit etre pris en compte."
890 PRINT " Precisez,pour chaque valeur,
si elle doit etre vraie."
900 PRINT " Reponses par oui ou non (o o
u n).":PRINT
910 PRINT TAB(34)"CONSULTATION"
920 PLOT 5,35:DRAW 635,35:DRAW 635,260:D
RAW 5,260:DRAW 5,35
930 x=0:wi=1
940 FOR i=1 TO n1
950   CLS#1:GOSUB 1190:' Valeurs choisie
s
960 NEXT i
970 nn=nvr:l=13:GOSUB 1360:vr$=nn$+vr$
980 CLS#1:PRINT #1, TAB(36)"REPONSES":PR
INT #1
990 IF nvr=0 THEN FOR k=1 TO n3:PRINT #1
,TAB(5);rs$(k):NEXT:GOTO 1140
1000 FOR j=1 TO nvr
1010   x=VAL(MID$(vr$,j*13+1,13))
1020   ns=VAL(LEFT$(suc$(x),13))
1030   FOR k=1 TO ns
1040     nvr1=0
1050     y=VAL(MID$(suc$(x),k*13+1,13))
1060     na=VAL(LEFT$(ant$(y),13))
1070     FOR k1=1 TO na
1080       z=VAL(MID$(ant$(y),k1*13+1,13
))
1090       IF bitv(z)=1 THEN nvr1=nvr1+1
1100     NEXT k1
1110     IF nvr1=nvr THEN GOSUB 1670:' L
iste des reponses
1120   NEXT k
1130 NEXT j
1140 wi=0
1150 RETURN
1160 '
1170 '        VALEURS CHOISIES
1180 '
1190 PRINT #1,"  Critere no.";i;": ";LEF
T$(cr$(i),11):PRINT #1
1200 qu$=" Ce critere doit-il etre pris
en compte ? ":GOSUB 1420:PRINT #1
1210 IF UPPER$(r$)="N" THEN RETURN
1220 FOR k1=1 TO nv(i)
1230   j=VAL(MID$(cr$(i),11+1+(k1-1)*13,
13))
```

```
1240    qu$=vl$(j)+" ? ":GOSUB 1420
1250    IF UPPER$(r$)="O" THEN bitv(j)=1:
nn=j:l=13:GOSUB 1360:vr$=vr$+nn$:nvr=nvr
+1:RETURN
1260 NEXT k1
1270 RETURN
1280 '
1290 '        LONGUEUR NORMALISEE A L CAR
. D'UNE CHAINE CH$
1300 '
1310 ch$=LEFT$(ch$+STRING$(l,CHR$(32)),l
)
1320 RETURN
1330 '
1340 '        LONGUEUR NORMALISEE D'UN NO
MBRE NN A L CAR. ET JUST. A DROITE
1350 '
1360 nn$=STR$(nn):nn$=MID$(nn$,2)
1370 nn$=RIGHT$(STRING$(l,CHR$(32))+nn$,
l)
1380 RETURN
1390 '
1400 '        REPONSE A UNE QUESTION QU$
PAR OUI OU NON
1410 '
1420 PRINT #wi, qu$;:r$=""
1430 WHILE UPPER$(r$)<>"O" AND UPPER$(r$
)<>"N"
1440    r$=INKEY$
1450 WEND
1460 PRINT #wi,r$:r=VAL(r$)
1470 RETURN
1480 '
1490 '      SUCCESSEURS DES VALEURS
1500 '
1510 l=13
1520 FOR k=1 TO n3
1530    nn=k:GOSUB 1360
1540    na=VAL(LEFT$(ant$(k),13))
1550    FOR k1=1 TO na
1560      j=VAL(MID$(ant$(k),k1*13+1,13))
1570      suc$(j)=suc$(j)+nn$:ns(j)=ns(j)
+1
1580    NEXT k1
1590 NEXT k
1600 FOR j=1 TO n2
1610    nn=ns(j):GOSUB 1360:suc$(j)=nn$+s
uc$(j)
1620 NEXT j
```

```
1630 RETURN
1640 '
1650 '        LISTE DES REPONSES
1660 '
1670 IF nreponses=0 THEN reponses$(1)=rs
$(y):nreponses=1:RETURN
1680 FOR k2=1 TO nreponses
1690    IF reponses$(k2)=rs$(y) THEN RETU
RN
1700 NEXT k2
1710 nreponses=nreponses+1:reponses$(nre
ponses)=rs$(y)
1720 RETURN
```

POUR ALLER PLUS LOIN

Les problèmes solubles par la famille "EXPERS" que nous venons de proposer se rencontrent fréquemment et peuvent être d'importance. Il en existe cependant de plus généraux. La gamme des possibilités peut ainsi être étendue dans trois directions au moins.

a) Les "opérateurs" logiques qui interviennent dans la base de connaissances ne se limitent pas à la conjonction ET. Les règles de cette base peuvent lier des propositions utilisant toutes combinaisons des opérateurs ET, OU, NON, IMPLIQUE, etc. L'expression logique la plus générale utilisant ces opérateurs peut d'ailleurs toujours être traduite, il est intéressant de le signaler, en une autre équivalente, contenant les seuls opérateurs ET et NON. Ce problème relève donc davantage de la "compilation" de la base de règles que du moteur d'inférence.

b) Le domaine des énoncés n'est pas forcément séparable en condition d'une part et réponses de l'autre. Considérons les deux règles suivantes :

– tout végétal à fleurs et graines est phanérogame ;
– tout végétal phanérogame et à graines nues est un sapin.

La propriété "phanérogame" est réponse dans la première proposition et condition dans la seconde.

c) Le domaine exploré peut dépasser les simples propositions et inclure la logique des prédicats ou logique du premier ordre.

L'analyse et la synthèse de tels systèmes experts plus généraux ne peut être présentée en quelques pages et dépasse le cadre du présent livre. Le lecteur intéressé pourra se référer à notre précédent ouvrage *Apple, logique et systèmes experts* (Éditions du P.S.I, 1985).

Il y trouvera développements complémentaires et autres logiciels d'application.

Mais tout notre discours et nos programmes se sont, jusqu'ici, référés à une "logique binaire" limitée aux seules catégories du VRAI et du FAUX. *Quid,* alors, du "peut-être", des "je ne sais pas", du probable, du possible, de l'incertain ?

Il existe enfin des domaines où les règles du jeu demeurent mal connues. Sur quelles bases pourrions-nous construire un système expert capable de prévoir le temps (météorologique) dans les jours à venir ? Est-il possible de proposer à la "machine" une liste d'exemples et de la laisser construire ses propres règles par auto-apprentissage ?

Nous n'avons pu nous résoudre à laisser de côté des sujets aussi passionnants. Rendez-vous au prochain (et dernier) chapitre.

Chapitre 9
DES SYSTÈMES EXPERTS D'UN AUTRE TYPE

INCERTITUDE ET PROBABILITÉ

Nos récentes excursions au royaume des systèmes experts furent placées sous le signe des certitudes. Tout élément recherché, appelé "réponse", offrait, de façon sûre, certaines caractéristiques, organisées en critères et valeurs ou, dit autrement, satisfaisait à une liste précise de conditions.

Par ailleurs, l'utilisateur était censé savoir parfaitement ce qu'il voulait, au point de répondre, par oui ou non, sans hésitation ni ambiguïté, à une liste de questions.

Dans la vie réelle, de telles circonstances ne sont pas forcément réalisées. Elles risquent de l'être encore moins dans les principaux domaines d'application des systèmes experts, domaines qui relèvent, nous l'avons dit, plus du savoir-faire et des "règles de l'art" que de la science organisée.

Il importe donc de pouvoir représenter et utiliser deux formes d'incertitudes : celles qui affectent les bases de connaissance et celles qui touchent les situations rencontrées. Nous décrirons d'abord, brièvement, les procédés utilisés, à cette fin, par deux systèmes connus, MYCIN et PROSPECTOR dans les champs, ô combien aléatoires, de la médecine et de la recherche minière. Nous construirons ensuite, avec vous, un moteur d'inférence assez puissant et général, EXBAYES. Ayant ainsi gagné le droit de nous distraire, nous traiterons, grâce à lui, avec le sourire, deux importants problèmes (parmi une infinité de possibles) :

- découvrir le tempérament caché de tout un chacun ;
- "poser un diagnostic" médical, sans prétention, pour distinguer rhumes, grippes... et autres laryngites.

La vraisemblance selon MYCIN

MYCIN traite les infections bactériennes du sang. En termes du chapitre précédent, qui lui conviennent tout à fait, ses réponses concernent l'identification d'organismes infectueux et les traitements adaptés. Ainsi :

- l'organisme est un pseudomonas ;
- je prescris de la clindamycine.

La base de règles (le savoir incorporé au système) et la base de faits (la situation donnée) sont structurées en objets (bactéries, cultures, antibiotiques, patients), critères et valeurs. Exemples :

Objets	*Critères*	*Valeurs*
Bactérie	Gram	Négatif
Bactérie	Forme	Bâtonnet
Culture	Site	Sang

Comme dans nos programmes EXPERS, chaque règle de la base exprime qu'une certaine réponse est conséquence de conditions simultanées et donc liées par la conjonction ET. Mais, première différence, la médecine n'étant pas une science exacte, à chaque règle est attaché un "coefficient de vraisemblance", donné par les experts. En voici un exemple :

SI le site de la culture est le sang
ET la coloration à GRAM négatif
ET la morphologie en bâtonnet
ET le patient un hôte à risques
ALORS il est probable (0,6) que l'organisme est un pseudomonas.

0,6 est ici le coefficient de vraisemblance attaché à la règle.

L'incertitude fondamentale touche les règles, nous venons de le voir, mais aussi les faits, qui sont donc également affectés de coefficients de vraisemblance. Ici apparaît un immense problème : comment composer toutes ces incertitudes afin d'évaluer celle relative à une réponse finale, obtenue par chaînage de faits et de règles ? La "recette" de MYCIN s'exprime ainsi :

- le CV (coefficient de vraisemblance) d'un nouveau fait, déduit d'une règle dont le CV propre est CR, est calculé par :

  $CV = CR * MIN(CV_i)$

  où CV_i désigne le CV de la i^e condition ;
- si deux règles différentes, dont les CV respectifs sont CV1 et CV2 donnent la même réponse, la CV de cette réponse est :

  CV réponse = CV1 + CV2 − CV1 * CV2

Notre propos n'est pas ici de présenter, en détail, la structure et les qualités de MYCIN, attestées par la satisfaction de ses usagers, mais, il

est difficile de ne pas le remarquer, les règles ci-dessus sont parfaitement arbitraires et rendraient nerveux n'importe quel statisticien. Nous pourrons, dans un instant, les comparer aux méthodes de PROSPECTOR, scientifiquement beaucoup plus solides. Les contextes, il est vrai, sont très différents.

Le risque principal, en médecine, serait de laisser échapper, au grand dam du patient, une maladie cachée et donc le traitement qui lui conviendrait. Il vaut mieux disposer d'une large gamme de diagnostics, quitte à prescrire quelques soins non indispensables (sans abuser de l'électrochoc ou de l'amputation). Dans cet esprit, MYCIN propose généralement une liste de causes possibles et de traitements, liste ordonnée selon des coefficients de vraisemblance qui n'exigent pas une grande précision.

Les situations traitées par PROSPECTOR sont à l'opposé. Il serait extrêmement coûteux de forer ou mettre en place de complexes et importantes installations dans tous les coins qui peuvent recéler de modestes trésors. Il faut ici pécher par défaut plutôt que par excès et affiner au maximum l'évaluation des probabilités.

La deuxième différence entre le moteur d'inférence de MYCIN et celui de nos programmes EXPERS est la technique d'exploration : MYCIN procède par "chaînage arrière", à partir d'une hypothèse vers les faits qui l'impliqueraient et oriente ainsi les questions à l'adresse de l'utilisateur. EXPERS part des conditions définies, en "chaînage avant", et détermine, de son propre chef, toutes les réponses qui y satisfont. Le programme EXBAYES, nous le verrons tout à l'heure, fonctionne dans les deux sens et choisit à tout moment une direction, selon les dernières informations reçues.

PROSPECTOR et le Révérend Bayes

Nous avons, supposons-le, des vues sur un site géographique tel que... le sous-sol de l'Ile-de-France, à la verticale de Notre-Dame, et une question dans la tête : y a-t-il, ou non, en ce lieu, un gisement intéressant de pétrole ? Nous voulons une réponse aussi précise que possible. Comment allons-nous procéder ?

D'abord, ce site n'a pas été choisi tout à fait au hasard. Compte tenu des informations générales disponibles, nous estimons, au départ, non nulle la probabilité d'y trouver du pétrole. Il doit même être possible d'évaluer, certes avec quelque approximation, la valeur, entre 0 et 1, de cette probabilité initiale, que nous appellerons P0.

Nous disposons ensuite d'experts (humains) et d'une longue liste d'indices (structure géologique, aspect de la végétation, indices magnétiques, gravimétriques, électriques, radio-actifs) dont nous pouvons, par observations sur le terrain, définir la présence (réponse Oui) ou l'absence (réponse Non). Supposons la campagne d'observations terminée et notre liste complètement remplie de Oui et Non. Qu'en

pouvons-nous conclure ? Que devient, dans ce nouveau contexte, notre probabilité initiale P0 ?

Si l'expérience passée, expérimentale ou théorique, nous montre qu'un indice I a été fréquemment relevé dans les gisements importants de pétrole déjà reconnus, alors, la présence d'un tel indice sera encourageante. En d'autres termes, elle accroît la probabilité initiale. Oui, mais de combien ? Et comment tenir compte de la totalité de ces effets, pour la gamme complète des indices ?

La réponse exige :

- une bonne documentation scientifique et statistique ;
- une pensée émue pour le Révérend Bayes qui, méditant sur les Idées éternelles, bien des années avant la découverte du pétrole, démontra le théorème qui porte son nom.

Avec la certitude de vous ennuyer, je dois être plus précis et avancer quelques notations et relations. Appelons H l'hypothèse (gisement intéressant de pétrole) que nous voulons valider. La probabilité initiale P0 se réfère à cette hypothèse. **Pour chaque indice,** nous devons, par tous les moyens, évaluer deux probabilités :

- la probabilité PV que l'indice I **soit effectivement observé si l'hypothèse H est vraie ;**
- la probabilité PF que l'indice I **soit effectivement observé si l'hypothèse H est fausse.**

La responsabilité d'établir les valeurs de PV et PF, pour chaque indice de la liste, revient aux experts (humains, pour l'instant du moins). Ces groupes de deux chiffres peuvent condenser une fantastique masse de connaissances. Ici intervient le Révérend. Deux cas possibles :

a) Selon les observations faites sur le gisement exploré, **l'indice I a, effectivement, été relevé.** La nouvelle probabilité P1 qui en résulte, pour l'hypothèse H, s'exprime par :

$$P1=P0*PV/(P0*PV+(1-P0)*PF)$$

b) Selon les mêmes observations, **l'indice I n'a pas été relevé.** P1 sera alors calculé par :

$$P1=P0*(1-PV)/(P0*(1-PV)+(1-P0)*(1-PF))$$

Il suffit, pour traiter tous les indices, d'itérer la procédure. Nous commençons par le premier indice et calculons P1, comme ci-dessus. Puis sont introduites les probabilités analogues à PV et PF, mais relatives au deuxième indice. P0 est remplacé, au second membre des formules, par la dernière valeur calculée P1. Les mêmes relations donnent P2 et ainsi de suite.

L'histoire s'arrêterait là s'il était toujours possible de répondre honnêtement, par oui ou non, à toute question de fait... Les concepteurs de PROSPECTOR ont songé aux affres des hommes de terrain obligés de répondre par oui ou non à des questions telles que (j'invente !) : "Y a-t-il abondance de pierres dichroïques ?" Ou encore : " Avez-vous

observé partout l'évolution lente des pechblendes en biotites ?" L'antidote est une échelle de valeurs, de −5 à +5, soit −5 pour Non et +5 pour Oui. Immense avantage : le zéro permet d'afficher, sans remords, l'ignorance absolue. Inconvénient : les très scrupuleux connaissent, avec une notable amplification, la situation de l'âne de Buridan.

Reste à traiter les valeurs ainsi obtenues. Nous pouvons, aux extrémités de l'échelle, afficher les correspondances :

Oui	+5	PV	PF
Non	−5	1−PV	1−PF

La solution pratique, adoptée par PROSPECTOR, est le calcul, pour chaque note, de probabilités intermédiaires, entre PV et 1−PV par exemple, grâce à une interpolation linéaire. Cette solution, rigoureuse aux bornes et au centre, pour la note 0 (dans ce dernier cas, elle laisse inchangée la probabilité initiale), semble donner ailleurs des résultats satisfaisants.

ABC de Probabilités

A seule fin de justifier le théorème de Bayes, qui joue un rôle essentiel dans les systèmes experts probabilistes, nous présentons ci-après les toutes premières notions de la théorie des probabilités. Vous avez le droit de sauter ce paragraphe si vous êtes expert en ce domaine ou si, tout simplement, vous préférez faire confiance.

Appelons "événement" n'importe quoi qui peut, ou non, survenir. Ce peut être, d'ailleurs, un ensemble d'autres événements. Supposons pouvoir associer à chaque événement d'une certaine classe un coefficient, compris entre 0 et 1, bornes incluses, dont le sens intuitif sera une probabilité d'existence. La probabilité d'un événement impossible est 0, celle d'un événement certain est 1.

Deux axiomes fondamentaux régissent ces probabilités. Soient E1 et E2 deux événements et P(E1), P(E2) leurs probabilités respectives. La non-réalisation d'un événement E sera désignée par NON E. Choisissons les notations suivantes :

- P(E1 ET E2) est la probabilité d'existence de l'un **ET** de l'autre événement ;
- P(E1 OU E2) est la probabilité d'existence de l'un **OU** de l'autre événement ;
- P(E2/E1) est la probabilité **conditionnelle** que E2 survienne **SI** E1 est déjà réalisé.

Alors P(E1 ET E2) = P(E1)*P(E2/E1) = P(E2)*P(E1/E2) dans tous les cas.

Si, d'autre part, E1 et E2 sont **exclusifs,** c'est-à-dire ne peuvent coexister, alors P(E1 OU E2) = P(E1) + P(E2).

Cas encore plus particulier : E1 et E2 sont complémentaires ou contradictoires. Ils s'excluent, mais l'un ou l'autre est obligatoirement réalisé. L'événement E1 OU E2 est certain. La probabilité associée est donc 1. Il s'ensuit :

P(E2) = 1 − P(E1)

Introduisons maintenant deux événements particuliers : une hypothèse H et un indice E. L'hypothèse sera l'existence d'un gisement important dans le cas de PROSPECTOR, un tempérament de type "passionné" pour un individu, la présence de la grippe chez un patient dans un diagnostic médical... ou toute autre "réponse" que nous attendons d'un système expert en activité. L'indice E sera un fait avéré, une observation, un symptôme, la réponse par oui ou non à une question... ou tout événement que nous avons précédemment classé parmi les "valeurs".

Nous supposons connue la probabilité "*a priori*" P(H) attachée à l'hypothèse H, avant toute information relative à l'indice E. Nous voulons calculer la nouvelle probabilité "*a posteriori*" P(H/E) de cette même hypothèse H si l'indice E est vérifié. Les relations précédentes nous permettent d'écrire :

P(E ET H) = P(E) * P(H/E) = P(H) * P(E/H)

D'où

P(H/E) = P(H) * P(E/H)/P(E)

D'autre part, les événements :

E1 = E ET H
E2 = E ET NON H

s'excluent mutuellement et leur union forme l'événement E. En d'autres termes :

P(E) = P(E1 OU E2) = P(E1) + P(E2)

ou

P(E) = P(H) * P(E/H) + P(NON H) * P(E/NON H)

Remarquons que P(NON H) = 1 − P(H) puis rapprochons ces diverses égalités. Nous obtenons l'expression, déjà écrite plus haut, du théorème de Bayes dans l'hypothèse où se vérifie l'existence de l'indice E auquel nous nous intéressons. Si E n'est pas réalisé, alors NON E est vrai, par définition, et les probabilités conditionnelles associées sont :

P(NON E/H) = 1 − P(E/H)

et

P(NON E/NON H) = 1 − P(E/NON H)

UN GÉNÉRATEUR DE SYSTÈMES EXPERTS PROBABILISTES

Le programme EXBAYES, puisqu'il faut l'appeler par son nom, est utilisable dans tous les domaines où vous pourriez construire une base de connaissances selon la structure qui va être bientôt décrite.

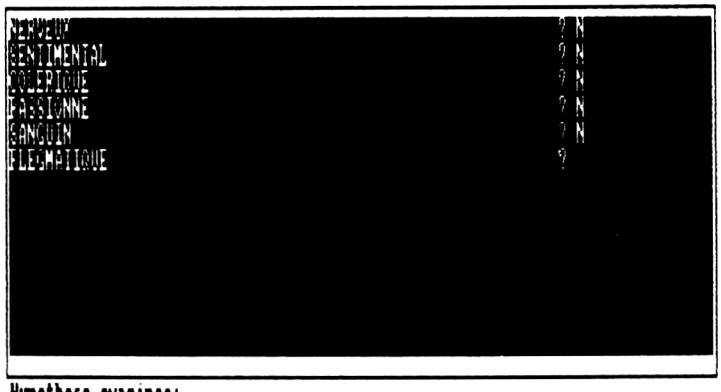
Prise de contact

Supposons EXBAYES disponible sur une de vos disquettes ou cassettes, ainsi que la base CARACT. Cet ensemble, convenablement interrogé, vous fera part de son opinion quant à votre tempérament. Chargez et lancez le programme. L'écran affiche ceci :

```
                    EXBAYES

 Nommez la base de connaissances que vous souhaitez interroger
parmi celles deja enregistrees sur disquette ou cassette.
 Vous indiquerez ensuite votre avis, par oui ou non (o ou n),
ou par une note (0 a 8) selon les questions qui vous seront posees.
 EXBAYES affichera les reponses convenables et les probabilites associees.

 Nom de la base? CARACT

          Travaux preparatoires sur la base

               Quelques secondes SVP.
```

A la question "Nom de la base ?", vous avez répondu CARACT, ce qui, après lecture (sur disquette ou cassette) du fichier correspondant et quelques secondes, appelle un nouvel écran. Il vous est demandé de choisir une hypothèse ou réponse possible parmi celles qui vont apparaître. Dans notre cas, elles représentent 8 classes de caractères ou tempéraments, allant de l'amorphe au passionné, à travers les nerveux, flegmatiques, colériques, etc. Toute réponse Non (soit N ou n) déclenche l'inscription de l'hypothèse suivante. Un Oui (O ou o) arrête le défilement et détermine l'hypothèse initiale de travail du programme.

```
Les hypotheses possibles vont defiler devant vous.
Par reponses Oui ou Non (O ou N), choisissez votre preferee.
Repondez ensuite par une note de 0 (pour NON) a 8 (pour OUI).

                    CONSULTATION
```




```
Hypothese examinee:
```

Ces préliminaires engendrent une liste de questions à chacune desquelles vous voudrez bien répondre par une note allant de 0 à 8. La note 0 signifie un Non définitif et catégorique. Un Oui de même sorte est associé à la note 8. Vous avez toute latitude pour adopter des positions moins tranchées, jusqu'à l'ignorance ou l'indifférence (note 4). Dans le cas général, vos réponses vont aiguiller le programme sur telle ou telle hypothèse. Celle sur laquelle il travaille à un instant donné et qui oriente ses questions apparaît au bas de l'écran.

```
Les hypotheses possibles vont defiler devant vous.
Par reponses Oui ou Non (O ou N), choisissez votre preferee.
Repondez ensuite par une note de 0 (pour NON) a 8 (pour OUI).

                    CONSULTATION
```

```
Hypothese examinee:          PASSIONNE
```

Lorsqu'il estime en savoir assez et qu'il s'est forgé une ou plusieurs opinions, EXBAYES affiche, sous le titre REPONSES, à l'intérieur d'un nouvel écran, la ou les réponses, affectées de probabilités. Par commodité, ces dernières sont ici exprimées en pourcentage mais elles pourraient l'être également en fractions décimales de l'intervalle (0,1).

Ne vous étonnez pas si la somme des probabilités affichées est supérieure à 1. Les hypothèses de la base ne sont, le plus souvent, nullement exclusives et leurs domaines de validité s'interpénètrent.

```
Les hypotheses possibles vont defiler devant vous.
Par reponses Oui ou Non (O ou N), choisissez votre preferee.
Repondez ensuite par une note de 0 (pour NON) a 8 (pour OUI).

                    CONSULTATION

                    REPONSES

COLERIQUE                                  74.68  %
PASSIONNE                                  63.87  %
```

Structure d'une base de connaissances

Leur forme généralise le cas, étudié au chapitre 8, des liaisons certaines. La base, écrite sous forme de DATA, comprend deux parties, respectivement associées aux "valeurs", c'est-à-dire aux faits ou indices et aux hypothèses, toujours qualifiées de "réponses". Nous avons en effet préféré conserver le même vocabulaire.

Les valeurs, encadrées par les mots "VALEURS" et "FIN VALEURS", contiennent seulement, pour chaque ligne de DATA :

- un numéro d'ordre entier ;
- le libellé d'une question.

Les données relatives aux réponses, délimitées par les mots "REPONSES" et "FIN REPONSES" sont plus complexes, comme le montre l'extrait, ci-dessous, de la base CARACT.

```
20000 DATA VALEURS
20010 DATA 1,Vous enthousiasmez vous ou
vous indignez vous aisement
20020 DATA 2,Supportez vous la critique
sans en etre blesse
20030 DATA 3,Etes vous angoisse devant u
ne tache nouvelle
20040 DATA 4,Etes vous stimule par les d
ifficultes
20050 DATA 5,Aimez vous travailler de fa
con reguliere
20060 DATA 6,Pensez vous que la ponctual
ite est une marque de faiblesse
20070 DATA 7,Envisagez vous toutes les p
ossibilites avant d'agir
20080 DATA 8,Pensez vous que le premier
mouvement est souvent le bon
20090 DATA 9,Ceux qui manient les abstra
ctions vous agacent-ils
20100 DATA FIN VALEURS
20110 DATA REPONSES
20120 DATA NERVEUX,.125,1,.9,.43,2,.1,.5
7,3,.9,.43,4,.1,.57,5,.1,.57,6,.9,.43,7,
.1,.57,8,.9,.43,9,.9,.43,*
20130 DATA SENTIMENTAL,.125,1,.9,.43,2,.
1,.57,3,.9,.43,4,.1,.57,5,.1,.57,6,.9,.4
3,7,.9,.43,8,.1,.57,9,.1,.57,*
```

A chaque réponse, au sein d'une ligne de DATA, sont associés :

- un libellé ;
- la probabilité *a priori,* exprimée en fraction décimale, attachée à la réponse ;
- une suite de triplets, en nombre égal au nombre de valeurs liées à la réponse considérée ; chaque triplet contient :

- le numéro d'identification de la valeur ;
- la probabilité conditionnelle que cette valeur soit constatée, si la réponse est vraie ;
- la probabilité conditionnelle que cette valeur soit constatée, si la réponse est fausse ;
- un astérisque terminal jouant le rôle de séparateur.

Soit, dans notre exemple, la ligne 20130. Elle contient :

- le mot "SENTIMENTAL" ;
- la probabilité *a priori* 0,125 ;
- le numéro de la question 1, soit "Vous enthousiasmez-vous ou vous indignez-vous aisément ?" ;
- la probabilité 0,9 que la réponse à cette question soit **Oui** si la personne interrogée **est** du type "SENTIMENTAL" ;
- la probabilité 0,43 que la réponse à cette question soit **Oui** si la personne interrogée **n'est pas** du type "SENTIMENTAL" ;
- les informations analogues pour les valeurs ou questions dont les numéros vont de 2 à 9 ;
- l'astérisque terminal.

Organisation du programme

Les premiers modules du programme EXBAYES ressemblent à ceux de EXPERS3. Après les messages de présentation, la base, choisie par l'utilisateur, est lue sur la mémoire de masse et réunie au programme principal.

Les lignes de DATA qui en résultent sont lues en deux passes. La première détermine les nombres NV de valeurs et NR de réponses afin de dimensionner les nombreuses tables du programme. La deuxième passe écrit les données dans les tables suivantes :

- libellés des valeurs : VL$(NV) ;
- libellés des réponses RS$(NR), probabilités initiales ou *a priori* PINIT(NR), chaînes de caractères ANT$(NR) contenant le nombre et les numéros d'ordre des valeurs qui sont "antécédents" de la réponse concernée (*cf.* chapitre 8), probabilités conditionnelles PROSIREV(NV, NR) et PROSIREF(NV, NR).

Comme vous l'aurez certainement remarqué, PROSIREV(J,K) voudrait signifier probabilité de réponse oui à la question (valeur) numéro J si la réponse numéro K est vraie, et de même pour PROSIREF(J,K), *mutatis mutandis.*

Est ensuite appelé le sous-programme SUCCESSEURS DES VALEURS, identique à celui qui apparaît dans EXPERS3. Sa mission : à partir de la table ANT$(NR), calculer, pour chaque valeur, les numéros des réponses qui lui "succèdent" dans le graphe des données et les ranger dans une table SUC$(NV). Il sera possible, ainsi, d'explorer très rapidement le graphe dans les deux sens, des valeurs vers les réponses ou inversement.

EXBAYES dessine ensuite, sur l'écran, la fenêtre affectée aux questions et réponses, puis entreprend sa tâche principale : poser des questions intelligentes et en tirer les conclusions. Trois sous-programmes s'en chargent : CONSULTATION, CALCUL DES PROBABILITÉS, PROBABILITÉS MAX.

Au cours de nécessaires initialisations (lignes 710 à 740), sont notamment mis à zéro des marqueurs, pour les valeurs, soit MARQVAL(NV), et les réponses, soit MARQREP(NR). Ils serviront à noter les questions et réponses déjà utilisées. Ensuite, le schéma général peut être ainsi décrit.

1. - Demander à l'utilisateur le choix d'une hypothèse de départ par réponses Non ou Oui en face des libellés de réponses qui défilent (lignes 750 à 850).

2. - Noter le numéro, soit MAK de l'hypothèse (réponse) choisie ainsi que sa probabilité PRMA, à cet instant (ligne 860).

3. - Afficher au bas de l'écran le libellé correspondant ; explorer les "antécédents" de l'hypothèse choisie et donc l'élément ANT$(MAK), afin de trouver la première question, relative à cette hypothèse et qui n'a pas encore été posée (lignes 870 à 920). Deux cas sont alors possibles.

3.a. - Une telle question, de numéro J, existe : elle est alors marquée, soit MARQVAL(J)=1, et déclenche l'appel au sous-programme CALCUL DES PROBABILITES. Celui-ci affiche la question VL$(J), enregistre la réaction R (chiffre entre 0 et 8) de l'utilisateur, recherche, grâce à l'élément SUC$(J), **les seules** hypothèses concernées par cette question et, pour chacune d'elles de numéro K, recalcule (théorème de Bayes), s'il y a lieu, la probabilité *a posteriori* PROB(K) qui en résulte. Il appelle alors le sous-programme PROBABILITE MAX. Ce dernier balaie toutes les hypothèses (de numéro K) qui n'ont pas encore épuisé les questions qui leur sont liées (le marqueur MARQREP(K) est toujours à 0). Il cherche, parmi celles-ci, **l'hypothèse de probabilité maximum,** enregistre son numéro dans la variable MAK, la probabilité associée dans la variable PRMA et, (enfin !), boucle sur le paragraphe *3.*

3.b. - Toutes les questions, relatives à l'hypothèse (de numéro MAK) en cours d'examen, ont déjà été posées. Une marque, soit MARQREP-(MAK)=1, et un numéro d'ordre (variable RANG) lui sont alors affectés. Sa probabilité *a posteriori* restera dorénavant inchangée. Elle est mémorisée dans la table PROBMA(RANG). Si une condition d'arrêt est vérifiée, le contrôle des opérations revient au programme principal pour affichage des réponses et de leurs probabilités. Sinon, il y a retour au paragraphe *3.* Les conditions d'arrêt peuvent prendre de multiples formes. Nous avons adopté (ligne 960) une combinaison liée au niveau de probabilité de la première hypothèse rencontrée qui soit complètement renseignée et au rang de la dernière hypothèse analysée.

Le programme affiche généralement, mais pas nécessairement, les réponses dans l'ordre décroissant de leurs probabilités.

Par construction, **il utilise, à la fois et de façon dynamique, entièrement guidée par les réponses de l'utilisateur, le "chaînage arrière" (d'une hypothèse vers les questions qu'elle déclenche) et le "chaînage avant" (d'une question vers les hypothèses qui lui sont liées).** Le balayage séquentiel des données est ainsi évité et l'efficacité du système substantiellement accrue.

```
680 '
690 '          CONSULTATION
700 '
710 CLS:PRINT TAB(33)"SYSTEME EXPERT":PR
INT
720 rang=0:prma0=0:prma=0
730 FOR j=1 TO nv:marqval(j)=0:NEXT
740 FOR k=1 TO nr:prob(k)=pinit(k):marqr
ep(k)=0:NEXT
750 PRINT " Les hypotheses possibles von
t defiler devant vous."
760 PRINT " Par reponses Oui ou Non (O o
u N), choisissez votre preferee."
770 PRINT " Repondez ensuite par une not
e de 0 (pour NON) a 8 (pour OUI)."
780 PRINT:PRINT TAB(34)"CONSULTATION":LO
CATE 1,24:PRINT " Hypothese examinee:"
790 PLOT 5,35:DRAW 635,35:DRAW 635,260:D
RAW 5,260:DRAW 5,35
800 CLS #1:wi=1:r$="":k=0
810 WHILE UPPER$(r$)<>"O"
820   k=k+1:ch$=rs$(k):l=12:GOSUB 1010:'
 Longueur normalisee d'une chaine
830   qu$=ch$+" ? ":GOSUB 1120:' Reponse
 par oui ou non
840 IF k=nr THEN 800
850 WEND
860 mak=k:prma=prob(k):CLS #1
870 LOCATE 30,24:PRINT STRING$(40,CHR$(3
2)):LOCATE 30,24:PRINT rs$(mak)
880 na=VAL(LEFT$(ant$(mak),13)):j1=0:dra
p=0
890 WHILE j1<na AND drap<>1
900   j1=j1+1:j=VAL(MID$(ant$(mak),j1*13
+1,13))
910   IF marqval(j)=0 THEN marqval(j)=1:
drap=1
920 WEND
930 IF drap=1 THEN GOSUB 1460:GOTO 870
940 IF rang=0 THEN prma0=prma
```

```
950 rang=rang+1:marqrep(mak)=1:reponses$
(rang)=rs$(mak):probma(rang)=prma
960 IF prma0>=0.9 OR (prma0>=0.7 AND ran
g>1) OR (prma0>=0.5 AND rang>2) OR rang>
=nr THEN wi=0:RETURN
970 GOSUB 1580:GOTO 870


1430 '
1440 '        CALCUL DES PROBABILITES
1450 '
1460 qu$=vl$(j)+" ? ":GOSUB 1210:' Repon
se par un chiffre entre 0 et 8
1470 ns=VAL(LEFT$(suc$(j),13))
1480 FOR k1=1 TO ns
1490   k=VAL(MID$(suc$(j),k1*13+1,13)):I
F marqrep(k)=1 OR prob(k)=0 OR prob(k)=1
 THEN 1520
1500   p=prob(k):pv=prosirev(j,k):pf=pro
siref(j,k):p1=(2*pv-1)*r/8+1-pv:p2=(2*pf
-1)*r/8+1-pf
1510   prob(k)=p*p1/(p*p1+(1-p)*p2)
1520 NEXT k1
1530 GOSUB 1580:' Probabilite max
1540 RETURN
1550 '
1560 '        PROBABILITE MAX
1570 '
1580 prma=0
1590 FOR k=1 TO nr
1600   IF marqrep(k)<>1 AND prob(k)>=prm
a THEN prma=prob(k):mak=k
1610 NEXT k
1620 RETURN
```

Le programme EXBAYES

Voici le listing complet du programme EXBAYES, précédé d'une liste des tables et variables utilisées. Viendront ensuite deux applications divertissantes (du moins je l'espère) à seule fin d'illustrer le mode d'emploi du système expert proposé.

Liste des variables

Tables :

ANT$(NR) : antécédents des réponses dans la base de connaissances.

MARQREP(NR) : marqueurs des réponses.
MARQVAL(NV) : marqueurs des valeurs.
NS(NV) : nombre de successeurs d'une valeur.
PINIT(NR) : probabilités *a priori* des réponses.
PROB(NR) : probabilités *a posteriori* des réponses.
PROBMA(NR+1) : probabilités des réponses sélectionnées pour l'affichage.
PROSIREV(NV,NR) : probabilité qu'une valeur d'indice J soit vraie si la réponse d'indice K est vraie.
PROSIREF(NV,NR) : probabilité qu'une valeur d'indice J soit vraie si la réponse d'indice K est fausse.
REPONSES$(NR+1) : libellés des réponses sélectionnées pour l'affichage.
RS$(NR) : libellés des réponses.
SUC$(NV) : successeurs des valeurs.
VL$(NV) : libellés des valeurs.

Variables simples :

A$: nom de la base de connaissances utilisée.
CH$: variable intermédiaire pour une chaîne.
DRAP : drapeau (0 ou 1) indiquant si toutes les questions relatives à une hypothèse ont été ou non utilisées.
I, J, J1, K, K1 : indices de boucles.
L : longueur d'une chaîne de caractères.
L2 : longueur normalisée d'une valeur.
L3 : longueur normalisée d'un nombre.
MAK : indice de l'hypothèse en cours d'examen.
NA : nombre des antécédents d'une réponse.
NN : indice courant.
NN$: chaîne de caractères associée à NN.
NR : nombre total des réponses.
NS : nombre des successeurs d'une valeur.
NV : nombre total des valeurs.
PRMA : probabilité maximum.
PRMA0 : probabilité *a posteriori* de la première hypothèse dont toutes les questions ont été utilisées.
QU$: libellé d'une question.
P, P1, P2, PV, PF : variables de travail désignant diverses probabilités.
R : nombre entré au clavier.
RANG : numéro d'ordre de sélection des hypothèses retenues.
R$: entrée d'un caractère au clavier.
WI : drapeau (0 ou 1) pour impression de l'une ou l'autre des fenêtres.
Z$: variable intermédiaire de travail.

Listing du programme EXBAYES

```
10 INK 0,0:INK 1,23:PAPER 0:PEN 1:MODE 2
20 CLS:PRINT TAB(36)"EXBAYES":PRINT
30 PRINT " Nommez la base de connaissanc
es que vous souhaitez interroger"
40 PRINT "parmi celles deja enregistrees
 sur disquette ou cassette."
50 PRINT " Vous indiquerez ensuite votre
 avis, par oui ou non (o ou n),"
60 PRINT "ou par une note (0 a 8) selon
les questions qui vous seront posees."
70 PRINT " EXBAYES affichera les reponse
s convenables et les probabilites associ
ees."
80 PRINT:INPUT" Nom de la base";a$
90 IF LEN(a$)>8 THEN PRINT " Nom de fich
ier trop long !":GOTO 80
100 CHAIN MERGE a$,110
110 PRINT:PRINT TAB(15)"Travaux preparat
oires sur la base ";a$
120 PRINT:PRINT TAB(20)"Quelques seconde
s SVP."
130 DEFINT a-o,q-z:l2=60:l3=3
140 '
150 '----- Initialisation et dimensions
160 '
170 nv=0:READ z$
180 WHILE UPPER$(z$)<>"FIN VALEURS"
190   READ z$:nv=nv+1
200 WEND
210 IF(nv MOD 2)<>1 THEN PRINT " Erreur
dans la base.":END
220 nv=(nv-1)/2
230 nr=0:READ z$
240 WHILE UPPER$(z$)<>"FIN REPONSES"
250   READ z$:IF z$="*" THEN nr=nr+1
260 WEND
270 DIM vl$(nv),ns(nv),suc$(nv),marqval(
nv),marqrep(nr),rs$(nr),ant$(nr),pinit(n
r),prob(nr),probma(nr+1),reponses$(nr+1)
,prosirev(nv,nr),prosiref(nv,nr)
280 RESTORE
290 '
300 '----- Lecture des valeurs
310 '
320 READ z$
330 FOR j=1 TO nv
```

```
340   READ nn
350   READ ch$:l=12:GOSUB 1010:' Longueu
r normalisee d'une chaine
360   vl$(nn)=ch$
370 NEXT
380 READ z$
390 '
400 '------ Lecture des reponses
410 '
420 READ z$:l=13
430 FOR k=1 TO nr
440   READ rs$(k):READ pinit(k):na=0
450   READ z$:IF z$<>"*" THEN nn=VAL(z$)
:GOSUB 1060:ant$(k)=ant$(k)+nn$:na=na+1:
READ prosirev(nn,k):READ prosiref(nn,k):
GOTO 450
460   nn=na:GOSUB 1060:ant$(k)=nn$+ant$(
k)
470 NEXT
480 READ z$
490 GOSUB 1300:' Successeurs des valeurs
500 '
510 ' ------ Appel du systeme expert
520 '
530 WINDOW #1,2,79,10,22:PAPER #1,1:PEN
#1,0
540 GOSUB 710:' Consultation
550 CLS #1:PRINT #1,TAB(35)"REPONSES":PR
INT #1:k=1
560 WHILE k<=rang AND probma(k)>=0.1
570   PRINT #1,TAB(5);reponses$(k);TAB(6
0);ROUND(probma(k)*100,2);" %"
580   k=k+1
590 WEND
600 LOCATE 1,24:PRINT STRING$(75,CHR$(32
)):LOCATE 1,24
610 PRINT:qu$=" On continue ? ":GOSUB 11
20:' Reponse par oui ou non
620 IF UPPER$(r$)="N" THEN PRINT " A vot
re service !":END ELSE 540
630 '
640 '
650 '        SOUS-PROGRAMMES
660 '
670 '
680 '
690 '        CONSULTATION
700 '
```

```
710 CLS:PRINT TAB(33)"SYSTEME EXPERT":PR
INT
720 rang=0:prma0=0:prma=0
730 FOR j=1 TO nv:marqval(j)=0:NEXT
740 FOR k=1 TO nr:prob(k)=pinit(k):marqr
ep(k)=0:NEXT
750 PRINT " Les hypotheses possibles von
t defiler devant vous."
760 PRINT " Par reponses Oui ou Non (O o
u N), choisissez votre preferee."
770 PRINT " Repondez ensuite par une not
e de 0 (pour NON) a 8 (pour OUI)."
780 PRINT:PRINT TAB(34)"CONSULTATION":LO
CATE 1,24:PRINT " Hypothese examinee:"
790 PLOT 5,35:DRAW 635,35:DRAW 635,260:D
RAW 5,260:DRAW 5,35
800 CLS #1:wi=1:r$="":k=0
810 WHILE UPPER$(r$)<>"O"
820   k=k+1:ch$=rs$(k):l=12:GOSUB 1010:'
 Longueur normalisee d'une chaine
830   qu$=ch$+" ? ":GOSUB 1120:' Reponse
 par oui ou non
840 IF k=nr THEN 800
850 WEND
860 mak=k:prma=prob(k):CLS #1
870 LOCATE 30,24:PRINT STRING$(40,CHR$(3
2)):LOCATE 30,24:PRINT rs$(mak)
880 na=VAL(LEFT$(ant$(mak),13)):j1=0:dra
p=0
890 WHILE j1<na AND drap<>1
900   j1=j1+1:j=VAL(MID$(ant$(mak),j1*13
+1,13))
910   IF marqval(j)=0 THEN marqval(j)=1:
drap=1
920 WEND
930 IF drap=1 THEN GOSUB 1460:GOTO 870
940 IF rang=0 THEN prma0=prma
950 rang=rang+1:marqrep(mak)=1:reponses$
(rang)=rs$(mak):probma(rang)=prma
960 IF prma0>=0.9 OR (prma0>=0.7 AND ran
g>1) OR (prma0>=0.5 AND rang>2) OR rang>
=nr THEN wi=0:RETURN
970 GOSUB 1580:GOTO 870
980 '
990 '        LONGUEUR NORMALISEE A L CAR.
 D'UNE CHAINE CH$
1000 '
1010 ch$=LEFT$(ch$+STRING$(l,CHR$(32)),l
)
```

```
1020 RETURN
1030 '
1040 '        LONGUEUR NORMALISEE D'UN NO
MBRE NN A L CAR. ET JUST. A DROITE
1050 '
1060 nn$=STR$(nn):nn$=MID$(nn$,2)
1070 nn$=RIGHT$(STRING$(l,CHR$(32))+nn$,
l)
1080 RETURN
1090 '
1100 '        REPONSE A UNE QUESTION QU$
PAR OUI OU NON
1110 '
1120 PRINT #wi, qu$;:r$=""
1130 WHILE UPPER$(r$)<>"O" AND UPPER$(r$
)<>"N"
1140   r$=INKEY$
1150 WEND
1160 PRINT #wi,r$
1170 RETURN
1180 '
1190 '      REPONSE A UNE QUESTION PAR UN
 CHIFFRE (ENTRE O ET 8)
1200 '
1210 PRINT #wi,qu$;:r$=" "
1220 WHILE ASC(r$)<48 OR ASC(r$)>56
1230   r$=INKEY$:IF r$="" THEN r$=" "
1240 WEND
1250 PRINT #wi,r$:r=VAL(r$)
1260 RETURN
1270 '
1280 '      SUCCESSEURS DES VALEURS
1290 '
1300 l=13
1310 FOR k=1 TO nr
1320   nn=k:GOSUB 1060
1330   na=VAL(LEFT$(ant$(k),13))
1340   FOR k1=1 TO na
1350     j=VAL(MID$(ant$(k),k1*13+1,13))
1360     suc$(j)=suc$(j)+nn$:ns(j)=ns(j)
+1
1370   NEXT k1
1380 NEXT k
1390 FOR j=1 TO nv
1400   nn=ns(j):GOSUB 1060:suc$(j)=nn$+s
uc$(j)
1410 NEXT j
1420 RETURN
1430 '
```

```
1440 '          CALCUL DES PROBABILITES
1450 '
1460 qu$=vl$(j)+" ? ":GOSUB 1210:' Repon
se par un chiffre entre 0 et 8
1470 ns=VAL(LEFT$(suc$(j),13))
1480 FOR k1=1 TO ns
1490    k=VAL(MID$(suc$(j),k1*13+1,13)):I
F marqrep(k)=1 OR prob(k)=0 OR prob(k)=1
 THEN 1520
1500    p=prob(k):pv=prosirev(j,k):pf=pro
siref(j,k):p1=(2*pv-1)*r/8+1-pv:p2=(2*pf
-1)*r/8+1-pf
1510    prob(k)=p*p1/(p*p1+(1-p)*p2)
1520 NEXT k1
1530 GOSUB 1580:' Probabilite max
1540 RETURN
1550 '
1560 '          PROBABILITE MAX
1570 '
1580 prma=0
1590 FOR k=1 TO nr
1600    IF marqrep(k)<>1 AND prob(k)>=prm
a THEN prma=prob(k):mak=k
1610 NEXT k
1620 RETURN
```

Quel est votre caractère ou celui de vos ami(e)s ?

Il existe de nombreuses typologies des tempéraments ou caractères. Nous vous proposons celle élaborée par le psychologue Heymans à l'Université de Groningue, entre 1906 et 1918. Elle distingue trois "dimensions" fondamentales :

- l'émotivité,
- l'activité,
- la polarité "primaire-secondaire".

Les deux premières caractéristiques se comprennent aisément. La troisième se réfère aux durées de réaction ou de préparation. Les caractères de type primaire réagissent sur le champ, avec vivacité... et puis oublient. Les autres sont plus lents, mais ce qui leur arrive retentit plus longuement dans leur esprit. Ils aiment planifier, réfléchir à l'avance.

La combinaison des valeurs extrêmes, positive (présence de la caractéristique) ou négative (son absence) de ces trois dimensions crée huit types de caractères ainsi définis :

a) **nerveux** : Emotif - Non Actif - Primaire (E-NA-P) ;
b) **sentimental** : Emotif - Non Actif - Secondaire (E-NA-S) ;

c) **colérique** : Emotif - Actif - Primaire (E-A-P) ;
d) **passionné** : Emotif - Actif - Secondaire (E-A-S) ;
e) **sanguin** : Non Emotif - Actif - Primaire (NE-A-P) ;
f) **flegmatique** : Non Emotif - Actif - Secondaire (NE-A-S) ;
g) **amorphe** : Non Emotif - Non Actif - Primaire (en abrégé NE-NA-P) ;
h) **apathique** : Non Emotif - Non Actif - Secondaire (NE-NA-S) ;

Les traits essentiels d'un caractère peuvent être déterminés par des questionnaires, dont chaque question vise à mettre en évidence l'intensité de telle ou telle dimension. Celui de Gaston Berger reste le plus connu. Si vous voulez en savoir davantage, consultez, par exemple les ouvrages suivants :

- *Connaître les autres,* par Michel Gauquelin (Denoël, 1970, pages 146 à 155),
- *Traité pratique d'analyse du caractère,* par Gaston Berger (P.U.F., 1963) ;
- *Questionnaire caractérologique pour l'analyse d'un caractère individuel,* par Gaston Berger (P.U.F., 1969).

Pour élaborer une base de connaissances, exploitable par notre programme EXBAYES, dans le domaine de la caractérologie, il convient de :

- disposer d'un questionnaire valable ;
- déterminer les probabilités *a priori* (ici les fréquences statistiques) relatives à chacun des huit types de base dans une population ;
- évaluer, pour chaque type et chaque question qui s'y rapporte, les probabilités conditionnelles, évoquées plus haut.

Comme il ne s'agit ici que d'illustration et non d'application en vraie grandeur, j'ai simplifié les données du problème. Le questionnaire de Gaston Berger contient 90 questions à réponse par Oui ou Non. J'ai ramené leur nombre à 9, soit 3 par dimension et les ai reformulées pour tenir compte de l'échelle numérique, de 0 à 8, des réponses possibles. Il vous sera loisible d'en ajouter.

Admettons par ailleurs que les huit types fondamentaux soient équitablement répartis dans la population globale. Leur probabilité *a priori* est donc 1/8, soit 0,125.

Restent les probabilités conditionnelles. Voyons cela sur un exemple. Si une personne est du type NERVEUX, son caractère contient une forte composante d'émotivité. Elle a donc toutes chances (mettons une probabilité 0,9 pour tenir compte des impondérables) de répondre Oui aux questions 1 et 3 (*cf.* la base donnée ci-dessous) et de faibles chances, (0,1 par exemple) de répondre Oui à la question 2. Si, par contre, cette personne n'est pas du type NERVEUX, selon l'hypothèse d'équirépartition des types, elle a une chance sur 7 d'appartenir à chacun des autres types et, dans cet ensemble, 3 types contiennent une composante d'émotivité. La probabilité conditionnelle de réponse Oui

aux questions 1 et 3 est donc 3/7, soit 0,43 et 4/7, soit 0,57 de répondre dans le même sens à la question 2. Des traitements similaires pour les autres types et questions produisent la base CARACT dont voici le texte.

```
20000 DATA VALEURS
20010 DATA 1,Vous enthousiasmez vous ou
vous indignez vous aisement
20020 DATA 2,Supportez vous la critique
sans en etre blesse
20030 DATA 3,Etes vous angoisse devant u
ne tache nouvelle
20040 DATA 4,Etes vous stimule par les d
ifficultes
20050 DATA 5,Aimez vous travailler de fa
con reguliere
20060 DATA 6,Pensez vous que la ponctual
ite est une marque de faiblesse
20070 DATA 7,Envisagez vous toutes les p
ossibilites avant d'agir
20080 DATA 8,Pensez vous que le premier
mouvement est souvent le bon
20090 DATA 9,Ceux qui manient les abstra
ctions vous agacent-ils
20100 DATA FIN VALEURS
20110 DATA REPONSES
20120 DATA NERVEUX,.125,1,.9,.43,2,.1,.5
7,3,.9,.43,4,.1,.57,5,.1,.57,6,.9,.43,7,
.1,.57,8,.9,.43,9,.9,.43,*
20130 DATA SENTIMENTAL,.125,1,.9,.43,2,.
1,.57,3,.9,.43,4,.1,.57,5,.1,.57,6,.9,.4
3,7,.9,.43,8,.1,.57,9,.1,.57,*
20140 DATA COLERIQUE,.125,1,.9,.43,2,.1,
.57,3,.9,.43,4,.9,.43,5,.9,.43,6,.1,.57,
7,.1,.57,8,.9,.43,9,.9,.43,*
20150 DATA PASSIONNE,.125,1,.9,.43,2,.1,
.57,3,.9,.43,4,.9,.43,5,.9,.43,6,.1,.57,
7,.9,.43,8,.1,.57,9,.1,.57,*
20160 DATA SANGUIN,.125,1,.1,.57,2,.9,.4
3,3,.1,.57,4,.9,.43,5,.9,.43,6,.1,.57,7,
.1,.57,8,.9,.43,9,.9,.43,*
20170 DATA FLEGMATIQUE,.125,1,.1,.57,2,.
9,.43,3,.1,.57,4,.9,.43,5,.9,.43,6,.1,.5
7,7,.9,.43,8,.1,.57,9,.1,.57,*
20180 DATA AMORPHE,.125,1,.1,.57,2,.9,.4
3,3,.1,.57,4,.1,.57,5,.1,.57,6,.9,.43,7,
.1,.57,8,.9,.43,9,.9,.43,*
```

```
20190 DATA APATHIQUE,.125,1,.1,.57,2,.9,
.43,3,.1,.57,4,.1,.57,5,.1,.57,6,.9,.43,
7,.9,.43,8,.1,.57,9,.1,.57,*
20200 DATA FIN REPONSES
```

Le poumon, vous dis-je !

Les applications médicales forment, nous l'avons vu, un domaine de prédilection des systèmes experts. Le programme EXBAYES, supposé muni d'une solide base de connaissances, offre une structure logique complète tout à fait adaptée au diagnostic médical. Ici encore, les étapes de construction de la base se définissent ainsi :

- liste des maladies qui pourront être reconnues ;
- fréquence relative de chacune d'elles parmi la population des consultants ;
- pour chaque maladie, liste des symptômes, accompagnée de leurs fréquences d'apparition, avec présence ou absence de la maladie considérée.

Le dernier point relève, en grande partie, de la séméiologie médicale, les compléments pouvant être apportés par l'expérience d'une équipe de médecins.

La création d'un tel outil sort nettement du cadre de ce livre. Montrer, sur une maquette, quel en serait le fonctionnement, ne semblait pas dépourvu d'intérêt. Dans cet esprit, et sans autre prétention, j'ai choisi une courte liste de neuf maladies des voies respiratoires et construit, sous contrôle médical, afin d'éviter les absurdités manifestes, la base "DOCTEUR", présentée ci-après. Son but, est-il besoin de le préciser, est de vous distraire, peut-être de vous instruire, mais en aucun cas de vous soigner.

```
20000 DATA VALEURS
20010 DATA 1,Avez vous de la fievre
20020 DATA 2,Avez vous des maux de tete
20030 DATA 3,Avez vous le nez qui coule
20040 DATA 4,Toussez vous beaucoup
20050 DATA 5,Votre voix s'est-elle modif
iee recemment
20060 DATA 6,Avez vous la gorge irritee
20070 DATA 7,Avez vous les yeux irrites
ou larmoyants
20080 DATA 8,Est ce que vous eternuez be
aucoup
20090 DATA 9,Avez vous la gorge seche
20100 DATA 10,Respirez vous avec difficu
lte
```

```
20110 DATA 11,Avez vous le nez bouche
20120 DATA 12,Avez vous eu une rhinite r
ecemment
20130 DATA 13,Vous sentez vous tres fati
gue
20140 DATA 14,Avez vous de la difficulte
 a avaler
20150 DATA 15,Avez vous des courbatures
20160 DATA 16,Sentez vous une douleur da
ns la poitrine
20170 DATA 17,X
20180 DATA 18,Avez vous des crises de su
ffocation
20190 DATA 19,Votre pouls bat-il vite
20200 DATA 20,Transpirez vous abondammen
t sans raison
20210 DATA 21,Etes vous souvent essoufle
20220 DATA 22,Avez vous une toux grasse
20230 DATA 23,Avez vous mal quand vous r
espirez ou toussez
20240 DATA 24,Etes vous en mauvaise sant
e depuis plusieurs semaines
20250 DATA 25,Anomalies constatees a l'o
ccasion d'efforts habituels
20260 DATA 26,Fumez vous
20270 DATA 27,Douleurs au visage ou au f
ront lorsque vous vous penchez
20280 DATA 28,Avez vous constate des gan
glions au cou
20290 DATA FIN VALEURS
20300 DATA REPONSES
20310 DATA RHINITE INFECTIEUSE,.02,8,.9,
.04,7,.8,.03,6,.8,.02,4,.5,.01,3,1,.01,2
,.2,.01,1,.5,.01,13,.8,.01,24,0,.01,*
20320 DATA SINUSITE,.01,12,.7,.01,11,.9,
.01,13,.8,.01,2,.6,.01,21,.3,.01,7,.5,.0
1,3,.6,.01,27,.9,.01,*
20330 DATA ANGINE,.02,6,1,.01,14,.9,.01,
1,.5,.01,9,.9,.01,26,.8,.04,28,.4,.01,*
20340 DATA GRIPPE,.01,6,.9,.01,8,.9,.01,
3,.5,.01,2,.8,.01,1,1,.01,13,1,.01,15,.8
,.01,16,.6,.01,24,0,.01,*
20350 DATA LARYNGITE,.01,5,1,.01,1,.6,.0
1,13,.05,.01,14,.7,.01,26,.8,.04,*
20360 DATA BRONCHITE AIGUE,.005,4,.8,.01
,1,1,.01,10,1,.01,13,1,.01,16,.5,.01,22,
1,.01,23,.9,.01,24,0,.01,21,.5,.01,*
```

```
20370 DATA BRONCHITE CHRONIQUE,.005,4,1,
.01,10,.9,.01,12,.5,.01,22,1,.01,21,.9,.
01,24,1,.01,25,.9,.01,26,.8,.4,*
20380 DATA ASTHME,.02,10,.8,.01,21,1,.01
,20,.5,.01,19,.5,.01,18,.6,.01,23,.8,.01
,*
20390 DATA EMPHYSEME,.01,21,1,.01,4,.001
,.01,10,.001,.01,22,.001,.01,26,.8,.4,*
20400 DATA FIN REPONSES
```

CLASSIFICATION ET APPRENTISSAGE

Nous avons, jusqu'ici, supposée connue la base de règles liant conditions et réponses dans les domaines explorés. Il n'en est pas forcément ainsi.

Pourrions-nous exprimer en forme de lois précises les relations entre le temps (météorologique) pour demain et les variables (pluie, vent, humidité, température, ensoleillement, etc.) des jours précédents ? Ou encore, sommes-nous capables d'écrire les liaisons, si elles existent, entre les traits de caractère d'une personne et son thème astrologique ?

Dans nombre de circonstances, nous disposons seulement d'une **liste d'exemples** et d'une autre **liste de classes** ou catégories. Chaque exemple décrit une situation particulière grâce aux valeurs prises par un certain nombre de variables d'état. Nous savons également affecter chaque situation de notre liste d'exemples à une classe définie. Mais, comme déjà dit, nous ne possédons pas de formule générale.

Or, et cela est très remarquable, une méthode d'intelligence artificielle, fondée sur des techniques statistique sophistiquées, va permettre à notre Amstrad de construire, par apprentissage, une telle formule générale. Nous proposerons à l'ordinateur, autant de fois que nécessaire, chaque exemple de notre liste et les valeurs correspondantes des variables d'état. Il devra deviner leur classe d'appartenance. S'il se trompe, nous rectifierons et il en prendra bonne note. La phase d'apprentissage sera terminée quand le programme sera capable de classer, sans erreur, toutes les situations de la liste. Le critère de sélection, qui aura alors été mis en place, déterminera à quelle classe correspond toute nouvelle situation du même type, c'est-à-dire définie par les mêmes variables d'état.

Vous avez peut-être gardé en mémoire le programme du même nom décrit en détail au chapitre 4. Grâce à lui, nous allons créer un nouveau système expert, baptisé EXCLASS. Il nous suffira :

- d'étendre la validité du programme à toutes les valeurs entières, positives, négatives ou nulles des variables d'état ; les variables logiques, VRAI ou FAUX, ne seront pas oubliées pour autant ; il suffira d'utiliser les représentations booléennes 1 et 0 ;

- d'écrire sur disquette ou cassette le fichier contenant la règle générale de classification bâtie par le programme au cours de son éducation ; elle nous aura coûté suffisamment d'efforts pour mériter d'être conservée ;
- de séparer les opérations d'apprentissage et d'utilisation ultérieure.

Comme tout système expert digne de ce nom, EXCLASS, vous le noterez, peut opérer en tous domaines et appeler, à tout instant, si vous le souhaitez, telle ou telle base de connaissances préalablement construite. Voyons, sur un exemple, ce que propose le programme.

La base CARASTRO

Ce nom, très mozartien, cache un ensemble de relations (hypothétiques) entre traits de caractère d'une personne et signe zodiacal de sa naissance. Il n'est pas nécessaire d'y croire pour trouver, à ce jeu, quelque intérêt. L'astrologie n'étant pas de ma compétence, j'ai consulté un ouvrage assez connu, *Les secrets du Zodiaque* de François-Régis Bastide, Bibliothèque du Club de la Femme, René Julliard et Editions Rombaldi, 1972. On y rencontre, pages 24 et 25, un tableau des qualités et défauts de chaque signe. Je vous le livre, ci-dessous, sans aucun engagement. J'avoue, d'ailleurs, que la note moyenne donnée, en matière de charme, aux natifs du Verseau, m'a profondément choqué. Vous trouverez le tableau, pages 262 à 265.

La signification des variables doit être claire. Chaque qualité ou défaut est noté entre −3 et +3 selon l'intensité supposée de ce trait de caractère. Par exemple, la note −3 pour la paresse indique un tempérament beaucoup plus travailleur que la moyenne. La note zéro correspond à la moyenne.

Le programme EXCLASS

Voyons ce que donne le programme sur l'exemple précédent. Nous devons, au départ, choisir entre les options proposées.

```
                                   EXCLASS

Options:
   1-Creation d'une base de connaissances
   2-Utilisation d'une base de connaissances
   3-Fin des operations
Votre choix ?
```

Il nous faut créer la base. Après avoir enregistré les critères (ici qualités et défauts) et les classes (signes zodiacaux), nous choisissons un signe zodiacal dans le tableau précédent et entrons les cotations pour chaque critère à mesure qu'ils défilent. Ceci fait pour les 21 critères, EXCLASS nous propose une classe. Comme pour le programme MEMOIRE, nous infirmons ou confirmons... et recommençons pour un autre signe zodiacal. Il faut, en moyenne, présenter deux à trois fois le tableau complet avant que le programme ne parvienne à éviter toute erreur.

Le temps est alors venu, et le programme en décide souverainement, d'écrire, sur la mémoire de masse, le fichier dont EXCLASS nous demande le nom (ici CARASTRO). Ce fichier comprend, dans l'ordre, les rubriques suivantes :

- nombres NC de critères et NR de réponses ;
- libellés CR$(NC) des critères et RE$(NR) des réponses ;
- table à deux dimensions COR(NC,NR) qui contient la matrice de sélection construite par le programme.

Tout est maintenant prêt pour une consultation qui promet d'être passionnante. Faites une liste des amis et relations dont vous connaissez, à la fois, le caractère et la date de naissance. Appelez l'option utilisation du programme, puis la base CARASTRO. Notez ensuite chaque personne selon les 21 critères, comme l'indique, par exemple, la copie d'écran ci-dessous.

```
                    EXCLASS

 Repondez, SVP, aux questions par une cotation
c'est dire un nombre entier, negatif, nul ou positif.
 Si vous desirez repondre par VRAI ou FAUX, utilisez 1 et 0.


Bonte                                   ? 1
Charme                                  ? 0
Courage                                 ? 1
Devouement                              ? 2
Double vue                              ? 0
Equilibre                               ? 0
Fidelite                                ? 0
Force                                   ? 1
Honnetete                               ? 2
Humour                                  ? 1
Idealisme                               ? 2
```

	Bélier	*Taureau*	*Gémeaux*
Acharnement	0	1	−1
Altruisme	0	0	0
Audace	3	0	1
Autorité	2	1	−3
Avarice	−3	3	−3
Bonté	0	1	−1
Charme	0	0	3
Courage	2	0	−2
Dévouement	0	−1	0
Double vue	−3	−2	0
Equilibre	−1	2	−3
Fidélité	0	2	−2
Force	2	1	−2
Honnêteté	2	1	−2
Humour	−1	0	3
Idéalisme	0	−2	−1
Impulsivité	3	−1	2
Orgueil	1	0	−2
Paresse	0	−3	1
Sens artistique	−1	1	1
Susceptibilité	0	2	0

	Cancer	*Lion*	*Vierge*
Acharnement	−2	0	1
Altruisme	0	1	0
Audace	−1	2	−3
Autorité	−2	3	−1
Avarice	−2	−3	2
Bonté	2	0	1
Charme	2	1	0
Courage	−2	3	−1
Dévouement	0	0	2
Double vue	3	−2	−2
Equilibre	−2	1	2
Fidélité	−2	0	3
Force	−1	3	1
Honnêteté	0	0	1
Humour	1	−3	2
Idéalisme	1	−1	−1
Impulsivité	2	1	−3
Orgueil	−1	3	−2
Paresse	3	−2	−3
Sens artistique	2	1	1
Susceptibilité	1	2	3

	Balance	*Scorpion*	*Sagittaire*
Acharnement	−1	0	0
Altruisme	1	0	2
Audace	0	1	2
Autorité	−2	0	1
Avarice	−1	−2	0
Bonté	2	0	2
Charme	1	2	−1
Courage	−2	0	2
Dévouement	1	0	1
Double vue	0	2	−1
Equilibre	3	−1	0
Fidélité	1	0	0
Force	−3	1	2
Honnêteté	0	−1	3
Humour	0	1	−1
Idéalisme	2	1	3
Impulsivité	−2	1	0
Orgueil	0	0	1
Paresse	1	0	0
Sens artistique	3	0	−3
Susceptibilité	−3	2	1

	Capricorne	*Verseau*	*Poisson*
Acharnement	3	0	−1
Altruisme	0	3	2
Audace	1	0	0
Autorité	2	0	−3
Avarice	3	−1	0
Bonté	−2	1	3
Charme	−3	0	1
Courage	−2	1	−1
Dévouement	−1	2	3
Double vue	−3	0	3
Equilibre	0	0	−2
Fidélité	1	0	3
Force	2	1	−2
Honnêteté	0	2	0
Humour	−3	1	−2
Idéalisme	2	2	3
Impulsivité	0	1	−2
Orgueil	1	−2	−3
Paresse	−2	0	−2
Sens artistique	−2	0	0
Susceptibilité	0	0	1

```
Impulsivite                                  ? 1
Orgueil                                      ? -2
Paresse                                      ? 0
Sens Artistique                              ? 0
Susceptibilite                               ? 0
```

Au terme du questionnaire, EXCLASS vous propose sa réponse, comme suit :

```
                                   EXCLASS

Nom de la base? CARASTRO
Repondez, SVP, aux questions par une cotation
c'est dire un nombre entier, negatif, nul ou positif.
Si vous desirez repondre par VRAI ou FAUX, utilisez 1 et 0.


Reponse: Verseau

Voulez vous continuer d'interroger la base CARASTRO ?
```

Vous seriez aimable de ne pas téléphoner à mon éditeur même si la réponse ne vous satisfait pas pleinement. Les mots désignant les qualités et défauts n'ont peut-être pas tout à fait le même sens pour le créateur du tableau et pour vous. Ou bien votre système de notation n'est pas vraiment impartial. Ou encore, mais ceci serait blasphématoire, les relations entre traits de caractère et signes zodiacaux sont purement imaginaires. Ou enfin ("il y a plus de choses sur terre et dans le ciel que n'en rêve notre philosophie") le signe zodiacal pourrait être corrigé par l'ascendant, les influences planétaires, "l'hérédité astrologique"... Que sais-je encore ?

De nombreuses possibilités s'offrent donc à vous, par exemple :

- revoir la liste des critères ;
- créer des classes plus complexes, prenant en compte certains des éléments évoqués ;
- constituer votre propre base de connaissances à partir d'exemples tirés de votre expérience et de vos relations et (le cas échéant) tester sa validité ;

- promouvoir un cabinet de "consultation astrologique avancée" utilisant EXCLASS et plusieurs bases dûment constituées (dans ce dernier cas, de substantielles royalties seraient bienvenues) ;
- généraliser à certains domaines très fréquentés tels les martingales, tiercés et autres lotos (même remarque).

Il convient, sans doute, de terminer cet ouvrage sur un ton moins badin. Le programme EXCLASS, quelques instants de réflexion vous en convaincront, présente l'intérêt de nous apprendre quelque chose. En échange d'un lot d'expériences et d'observations brutes, il nous rend une généralisation de celles-ci, une formule qui les synthétise... et dont il conserve les secrets de fabrication.

Il pourrait, de plusieurs façons, engendrer des versions plus élaborées. Voici quelques axes possibles de développement :

- mettre les exemples en forme de DATA et organiser le programme, dans la phase de constitution d'une base, afin qu'il les relise, de lui-même, jusqu'au terme de son éducation (les natifs du Cancer y avaient sûrement déjà pensé) ;
- ajouter des options afin de mémoriser la liste des exemples et de mettre à jour celle-ci (additions, suppressions, modifications) ;
- élaborer un critère d'efficacité des questions qui permette au programme de les choisir, en phase de consultation, selon les réponses déjà données et d'en réduire ainsi le nombre.

Il est temps de vous présenter le listing d'EXCLASS. La liste des variables est celle de MEMOIRE avec seulement deux ajouts :

- A$: nom de la base courante ;
- DRAP : drapeau qui indique la fin de la période d'apprentissage.

```
10 ' -------- Initialisation
20 INK 0,0:INK 1,23:PAPER 0:PEN 1
30 MODE 2:WINDOW #1,2,79,9,24
40 DEFINT a-z
50 CLS:CLEAR:PRINT TAB(36)"EXCLASS":PRIN
T
60 PRINT " Options:"
70 PRINT TAB(5)"1-Creation d'une base de
 connaissances"
80 PRINT TAB(5)"2-Utilisation d'une base
 de connaissances"
90 PRINT TAB(5)"3-Fin des operations"
100 PRINT " Votre choix ? ";:GOSUB 1290:
' Reponse par un chiffre
110 ON r GOTO 130,380,580
120 ' ----- Creation d'une base de conna
issances
```

```
130 GOSUB 650:' Entree des criteres et d
es reponses
140 CLS:CLS #1:PRINT TAB(36)"EXCLASS":PR
INT
150 PRINT " Je ne demande qu'a apprendre
. Pour chaque situation envisagee"
160 PRINT "indiquez moi, pour chaque cri
tere de la liste ci-dessous"
170 PRINT "la cotation qui convient (nom
bre entier negatif, nul ou positif)."
180 PRINT " Si vous desirez repondre par
 VRAI ou FAUX, utilisez 1 et 0."
190 PRINT " Je proposerai une reponse et
 vous m'indiquerez si elle est correcte.
"
200 GOSUB 790:' Valorisation des critere
s
210 GOSUB 940:' Test des reponses
220 IF drap=0 THEN 200
230 CLS:CLS #1
240 PRINT " Apprentissage termine."
250 PRINT " La base de connaissances va
etre ecrite sur cassette ou disquette."
260 INPUT" Nom de la base";a$
270 IF LEN(a$)>8 THEN 260
280 OPENOUT a$:PRINT #9,nc:PRINT #9,nr
290 FOR i=1 TO nc:PRINT #9,cr$(i):NEXT
300 FOR i=1 TO nr:PRINT #9,re$(i):NEXT
310 FOR i=1 TO nc
320   FOR j=1 TO nr
330     PRINT #9,cor(i,j)
340   NEXT j
350 NEXT i
360 CLOSEOUT:GOTO 50
370 ' ----- Utilisation d'une base de co
nnaissances
380 CLS:CLS #1:PRINT TAB(36)"EXCLASS":PR
INT
390 INPUT " Nom de la base";a$
400 OPENIN a$:INPUT #9,nc,nr
410 DIM cr$(nc),vlc(nc),re$(nr),rang(nr)
,cor(nc,nr)
420 FOR i=1 TO nc:INPUT #9,cr$(i):NEXT
430 FOR i=1 TO nr:INPUT #9,re$(i):NEXT
440 FOR i=1 TO nc
450   FOR j=1 TO nr
460     INPUT #9,cor(i,j)
470   NEXT j
480 NEXT i
```

```
490 CLOSEIN
500 PRINT " Repondez, SVP, aux questions
 par une cotation"
510 PRINT "c'est dire un nombre entier,
negatif, nul ou positif."
520 PRINT " Si vous desirez repondre par
 VRAI ou FAUX, utilisez 1 et 0."
530 GOSUB 790:' Valorisation des critere
s
540 CLS #1:PRINT #1," Reponse:  "; re$(j
p)
550 PRINT #1:PRINT #1," Voulez vous cont
inuer d'interroger la base ";:PRINT #1,a
$;:PRINT #1," ?";:GOSUB 1220:' Reponse o
ui ou non
560 IF LOWER$(r$)="o" THEN CLS:CLS #1:PR
INT TAB(36)"EXCLASS":PRINT:GOTO 500 ELSE
 50
570 ' ----- Fin des operations
580 END
590 '
600 '        SOUS-PROGRAMMES
610 '
620 '
630 '        ENTREE DES CRITERES ET DES R
EPONSES
640 '
650 CLS #1:PRINT #1,"Entrez les criteres
 qui doivent conditionner mes reponses."
:PRINT #1
660 INPUT #1,"Nombre de criteres";nc:DIM
 cr$(nc),vlc(nc):PRINT #1
670 FOR i=1 TO nc
680   PRINT #1,"Critere No.";i;:INPUT #1
,cr$(i)
690 NEXT i
700 CLS #1:PRINT #1,"Entrez les reponses
 possibles.":PRINT #1
710 INPUT #1,"Nombre de reponses";nr:DIM
 re$(nr),rang(nr),vr(nr),cor(nc,nr):PRIN
T #1
720 FOR i=1 TO nr
730   PRINT #1,"Reponse No.";i;:INPUT #1
,re$(i)
740 NEXT i
750 RETURN
760 '
770 '        VALORISATION DES CRITERES
780 '
```

```
790 CLS #1:maxr=0
800 FOR i=1 TO nc
810   PRINT #1,TAB(5);cr$(i);TAB(60):INP
UT #1,vlc(i)
820 NEXT i
830 FOR j=1 TO nr
840   rang(j)=0
850   FOR i=1 TO nc
860     rang(j)=rang(j)+vlc(i)*cor(i,j)
870   NEXT i
880   IF rang(j)>=maxr THEN maxr=rang(j)
:jp=j
890 NEXT j
900 RETURN
910 '
920 '        TEST DES REPONSES
930 '
940 PRINT #1:PRINT #1," Reponse '";re$(j
p);"'  Est-elle correcte ?";:GOSUB 1220:
'Oui ou non
950 ' -------- Reponse correcte
960 IF LOWER$(r$)="o" THEN vr(jp)=1:GOTO
 1130
970 ' -------- Reponse erronee
980 CLS #1
990 PRINT #1,"Voici la liste des reponse
s."
1000 PRINT #1,"Indiquez le numero de la
reponse correcte.":PRINT #1
1010 FOR j=1 TO nr
1020   vr(j)=0
1030   PRINT #1,j;":- ";re$(j)
1040 NEXT j:PRINT #1
1050 INPUT #1," Numero correct";jc
1060 FOR j=1 TO nr
1070   IF rang(j)>=maxr AND j<>jc THEN F
OR i=1 TO nc:cor(i,j)=cor(i,j)-vlc(i):NE
XT i
1080 NEXT j
1090 FOR i=1 TO nc
1100   cor(i,jc)=cor(i,jc)+vlc(i)
1110 NEXT i
1120 ' -------- Test de fin d'apprentissa
ge
1130 score=0:drap=0
1140 FOR j=1 TO nr
1150   score=score+vr(j)
1160 NEXT j
1170 IF score=nr THEN drap=1
```

```
1180 RETURN
1190 '
1200 '        OUI OU NON
1210 '
1220 r$=INKEY$
1230 IF r$="" OR (LOWER$(r$)<>"o" AND LO
WER$(r$)<>"n") THEN 1220
1240 PRINT #1,r$
1250 RETURN
1260 '
1270 '        REPONSE A UNE QUESTION PAR
UN CHIFFRE (ENTRE 1 ET 3)
1280 '
1290 r$=""
1300 WHILE r$<"1" OR r$>"3"
1310   r$=INKEY$
1320 WEND
1330 r=VAL(r$):PRINT r
1340 RETURN
```

Annexe
DISQUETTE D'ACCOMPAGNEMENT

Il vous sera facile, en utilisant, par exemple, le bon de commande ci-dessous, de recevoir une disquette, complément naturel de cet ouvrage. Le catalogue (copie d'écran) que voici en liste le contenu.

ALBE	.BAS	2K	EXPERS3	.BAS	6K
ASHBY	.BAS	2K	HARDCOPY	.	1K
AWELE1	.BAS	5K	JEU1	.BAS	2K
AWELE2	.BAS	6K	JEU2	.BAS	2K
AWELEVOL	.BAS	8K	LANGAGES	.	1K
CARACT	.	2K	LOTERIE	.BAS	3K
CARASTRO	.	2K	MEMOIRE	.BAS	3K
CELLULES	.BAS	2K	MINIMAX	.BAS	2K
CHIFFRES	.BAS	4K	NARCISSE	.BAS	7K
CYBER	.BAS	3K	OGRE	.BAS	2K
DATEAUX	.	2K	OISILLON	.	1K
DATLANG	.	1K	OPTIMUM	.BAS	1K
DOCTEUR	.	3K	PAT	.BAS	3K
EAUX	.	2K	SHANNON	.BAS	2K
EXBAYES	.BAS	6K	TAQUIN1	.BAS	4K
EXCLASS	.BAS	5K	TAQUIN2	.BAS	4K
EXPERS1	.BAS	4K	TAQUIN3	.BAS	5K
EXPERS2	.BAS	6K	ZERO	.BAS	1K

Cette disquette contient :

a) 27 programmes en BASIC Amstrad, construits, analysés et commentés au long des différents chapitres. Ils se distinguent par l'extension ".BAS" ;

b) 8 "bases de connaissances" qui peuvent être, parmi beaucoup d'autres, mises en œuvre par les systèmes experts présentés dans l'ouvrage.

Ces bases se divisent elles-mêmes en deux classes :

1. – Fichiers-programmes formés d'instructions DATA. Ils fusionnent avec les programmes principaux lorsqu'ils sont appelés par ceux-ci (instructions CHAIN MERGE). Pour cette raison, il est préférable, si vous les modifiez, de sauver, sur disquette, les nouvelles versions en "mode texte" (instruction SAVE "NOM",A). DATEAUX (vacances nautiques) et DATLANG (langages de programmation) sont adaptées aux programmes EXPERS1 et EXPERS3. Les bases CARACT (analyse du tempérament d'une personne) et DOCTEUR (diagnostic médical) font usage de probabilités et travaillent avec EXBAYES.

2. – Fichiers séquentiels : tel est le cas des fichiers LANGAGES et EAUX, dont les contenus sont analogues à ceux de DATLANG et DATEAUX, mais dont la forme est adaptée à EXPERS2 qui a permis d'ailleurs de les créer. Les bases CARASTRO (liens entre traits de caractère et thème zodiacal) et OISILLON (l'éducation d'un petit oiseau) sont également de ce type, mais travaillent avec le système expert EXCLASS, capable d'apprentissage.

Il vous sera dans doute agréable de trouver sur la disquette d'accompagnement le programme HARDCOPY responsable, sur imprimante EPSON, des copies d'écran qui illustrent ce livre.

Vente par correspondance

BON DE COMMANDE

Je commande la disquette d'accompagnement du livre : *Trois étapes vers l'intelligence artificielle sur Amstrad* au prix de 150,00 FF *.

Renvoyez-nous ce bon rempli (découpé ou copié), avec votre règlement, plus de 10,00 FF** de frais de port et d'emballage, par chèque bancaire ou postal établi à l'ordre de LA CONSOLE.
Adresse : La Console, 5, place du Colonel Fabien, 75010 Paris

Nom ..... Prénom .....
Adresse .....
..... Ville .....
Code postal ..... Pays .....

* Prix valable jusqu'au 31 décembre 1986.
** Hors CEE, prévoyez 15,00 FF pour supplément de frais d'expédition.

Conseils de lecture

Pour approfondir vos connaissances en BASIC, mieux connaître le système des CPC 464, 664 et 6128 et découvrir d'autres domaines de l'intelligence artificielle, P.S.I. vous propose une palette d'ouvrages utiles.

Pour maîtriser le BASIC Amstrad

☐ **Basic Amstrad 1 – Méthodes pratiques** – Jacques Boisgontier et Bruno Césard (Éditions du P.S.I.)

Pour ceux qui ont déjà pratiqué un BASIC, voici un ouvrage de perfectionnement au BASIC Amstrad. Un chapitre sur le CP/M 2.2 et le CP/M Plus donne les principales commandes systèmes.

☐ **Basic Amstrad 2 – Programmes et fichiers** – Jacques Boisgontier (Éditions du P.S.I.)

Pour pratiquer le Basic Amstrad, cet ouvrage donne de nombreux programmes de gestion, d'éducation et de jeu où le rôle des fichiers est expliqué et largement commenté.

☐ **Basic plus – 80 routines sur Amstrad** – Michel Martin (Éditions du P.S.I.).

Pour pousser votre Amstrad au maximum de ses capacités : 80 routines de simulation d'instructions qui n'existent pas en BASIC Amstrad.

Pour mieux connaître le système des CPC et du PCW 8256

☐ **Clefs pour Amstrad 1 – système de base** – Daniel Martin (Éditions du P.S.I.)

Mémento présentant synthétiquement le jeu d'instructions du Z80, les points d'entrée des routines système, les connecteurs et brochages, etc. Le livre de chevet du programmeur sur Amstrad.

☐ **Clefs pour Amstrad 2 – système disque** – Daniel Martin et Philippe Jadoul (Éditions du P.S.I.)

Ce deuxième tome consacré au système disque présente les points d'entrée des routines disque, les blocs de contrôle, la programmation et les brochages des circuits spécialisés...
La deuxième partie du livre est aussi destinée aux possesseurs d'Amstrad 8256.

☐ **CP/M Plus sur Amstrad** – Yvon Dargery (Éditions du P.S.I.)

Toutes les commandes CP/M et CP/M Plus pour maîtriser le système des 6128 et 8256 : un ouvrage de référence illustré par de nombreux programmes.

☐ **Le livre de l'Amstrad, Tome 1** – Daniel Martin et Philippe Jadoul (BCM – diffusé par P.S.I.)

Ce livre, destiné aux programmeurs des CPC 464 et 664, donne une étude complète de tous les circuits internes, et analyse la structure interne du BASIC. Vous y trouverez, en outre, une étude complète des RSX et des programmes de scrolling, de traçage de rectangles, de coloriage de surface et de manipulation vectorielle.

Pour découvrir d'autres domaines de l'intelligence artificielle

☐ **Apple, logique et systèmes experts** – René Descamps (Éditions du P.S.I.)

Une excellente introduction à la logique, illustrée par des programmes en BASIC Applesoft accessibles à tous. Trois microsystèmes experts permettent de concevoir ce qu'il est possible de réaliser dans le domaine professionnel sur de gros ordinateurs.

☐ **Systèmes industriels d'intelligence artificielle** – Lucas Pun (Editests, diffusé par P.S.I.)

Cet ouvrage d'adresse aux ingénieurs industriels, automaticiens et informaticiens qui ont à concevoir et à réaliser des Systèmes Industriels d'Intelligence Artificielle (SIIA).

A paraître :

☐ **Intelligence artificielle : langages et formes** – Thierry Lévy-Abegnolli et Olivier Magnan (Éditions du P.S.I.)

Voici deux autres grands domaines de l'intelligence artificielle, abordés ici sur Amstrad CPC, avec des programmes BASIC de haut niveau. Reconnaissance des formes et du langage, microsystème expert médical et étude des jeux de réflexion vous permettront de maîtriser encore davantage les multiples facettes de l'intelligence artificielle.

Votre avis nous intéresse

Pour nous permettre de faire de meilleurs livres, adressez-nous vos critiques sur le présent ouvrage.

— Ce livre vous donne-t-il toute satisfaction ?

.....

.....

.....

.....

.....

— Y a-t-il un aspect du problème que vous auriez aimé voir abordé ?

.....

.....

.....

.....

.....

Si vous souhaitez des éclaircissements techniques, écrivez-nous, nous ne manquerons pas de vous répondre directement.

Où avez-vous acheté ce livre ?

☐ cadeau ☐ librairie ☐ autres
☐ exposition ☐ boutique micro

Comment en avez-vous eu connaissance ?

☐ publicité ☐ catalogue ☐ autres
☐ exposition ☐ conseils d'un ami

Avez-vous déjà acquis des livres P.S.I. ?

Lesquels ? .....

.....

qu'en pensez-vous ? .....

.....

.....

Nom ..... Prénom ..... Age.....
Adresse .....
Profession .....
Centre d'intérêt .....

CATALOGUE GRATUIT

Vous pouvez obtenir un catalogue complet des ouvrages PSI, sur simple demande, ou en retournant cette page remplie à votre libraire, à votre boutique micro ou aux

Editions du PSI
BP 86
77402 Lagny-sur-Marne Cedex

Achevé d'imprimer par l'imprimerie
Tardy Quercy S.A. à Bourges
D. L. : Mars 1986 - N° d'Imprimeur : 12856
N° d'Édition : 86595-278-1

P.S.I.

MICRO ET TECHNIQUE

TROIS ÉTAPES VERS L'INTELLIGENCE ARTIFICIELLE

POUR AMSTRAD CPC

L'Intelligence Artificielle, ce domaine fascinant où se mêlent science et science-fiction, est une des clés de l'avenir. Ce livre en dévoile les ressorts de façon simple et pratique, grâce à des dizaines d'exemples et vingt-sept programmes en BASIC qui utilisent toutes les ressources de l'AMSTRAD.

Votre micro-ordinateur résoudra les puzzles du TAQUIN, simulera un pilote automatique, imitera les capacités d'adaptation et d'apprentissage des êtres vivants. Vous lui enseignerez les stratégies gagnantes des jeux de réflexion (échecs, dames, awele, etc.). Partenaire redoutable, il sera capable de jouer contre lui-même et d'améliorer sa technique à votre contact.

Les Systèmes Experts sont à la mode. Créez les vôtres et utilisez-les.

EXPERS, EXBAYES, EXCLASS sont capables de prendre en compte vos incertitudes ou de tirer des conclusions à partir d'exemples concrets. Ils vous aideront à préparer vos vacances, à deviner le tempérament de vos amis, à préciser les liens entre signe astrologique et traits de caractère. Ils vous proposent même une consultation médicale sans frais. Tout cela, bien sûr, avec humour !

9 782865 952786

ÉDITIONS DU P.S.I.
BP 86 – 77402 LAGNY S/MARNE CEDEX – FRANCE

ISBN : 2 86595-278 9

160 FF

ILLUSTRATION : LUC ARRIGNON

Document numérisé
avec amour par :
AMSTRAD
CPC
MÉMOIRE ÉCRITE
ACME
https://acpc.me/